Computational Learning Theory **Spring Semester, 2003/4**

Lecture 1: March 2

Lecturer: Yishay Mansour *Scribe: Gur Yaari, Idan Szpektor*

1.1 Introduction

Several fields in computer science and economics are focused on the analysis of Game theory. Usually they observe Game Theory as a way to solve optimization problems in systems where the participants act independently and their decisions affect the whole system.
Following is a list of research fields that utilize Game Theory:

- Artificial Intelligence (AI) - Multiple Agents settings where the problem is usually a cooperation problem rather than a competition problem.
- Communication Networks - Distribution of work where each agent works independantly.
- Computer Science Theory - There are several subfields that use Game Theory:
  - Maximizing profit in bidding
  - Minimum penalty when using distributional environment
  - Complexity
  - Behavior of large systems

1.2 Course Syllabus

- Basic definitions in Game Theory, concentrating on Nash Equilibrium
- Coordination Ratio
  - Comparison between global optimum and Nash Equilibrium
  - Load Balancing Models
- Computation of Nash Equilibrium
  - Zero Sum games (Linear Programming)
  - Existence of Nash Equilibrium in general games

VISIT…

- Regret - playing an "unknown" game. Optimizing a player's moves when the player can only view her own payoff
- Vector Payoff - the Payoff function is a vector and the target is to reach a specific target set
- Congestion and Potential games - games that model a state of load
- Convergence into Equilibrium
- Other...

1.3 Strategic Games

A strategic game is a model for decision making where there are N players, each one choosing an action. A player's action is chosen just once and cannot be changed afterwards.

Each player i can choose an action a_i from a set of actions A_i. let A be the set of all possible action vectors $\times_{j \in N} A_j$. Thus, the outcome of the game is an action vector $\vec{a} \in A$.

All the possible outcomes of the game are known to all the players and each player i has a preference relation over the different outcomes of the game: $\vec{a} \preceq_i \vec{b}$ for every $\vec{a}, \vec{b} \in A$. The relation stands if the player prefers $\vec{b}$ over $\vec{a}$, or has equal preference for either.

Definition A **Strategic Game** is a triplet $\langle N, (A_i), (\preceq_i) \rangle$ where N is the number of players, A_i is the finite set of actions for player i and $\preceq_i$ is the preference relation of player i.

We will use a slightly different notation for a strategic game, replacing the preference relation with a payoff function $u_i : A \to \mathbb{R}$. The player's target is to maximize her own payoff. Such strategic game will be defined as: $\langle N, (A_i), (u_i) \rangle$.

This model is very abstract. Players can be humans, companies, governments etc. The preference relation can be subjective evolutional etc. The actions can be simple, such as "go forward" or "go backwards", or can be complex, such as design instructions for a building.

Several player behaviors are assumed in a strategic game:

- The game is played only once
- Each player "knows" the game (each player knows all the actions and the possible outcomes of the game)
- The players are rational. A rational player is a player that plays selfishly, wanting to maximize her own benefit of the game (the payoff function).
- All the players choose their actions simultaneously

1.4 Pareto Optimal

An outcome $\vec{a} \in A$ of a game $\langle N, (A_i), (u_i) \rangle$ is Pareto Optimal if there is no other outcome $\vec{b} \in A$ that makes every player at least as well off and at least one player strictly better off. That is, a Pareto Optimal outcome cannot be improved upon without hurting at least one player.

Definition An outcome $\vec{a}$ is **Pareto Optimal** if there is **no** outcome $\vec{b}$ such that $\forall_{j \in N}\, u_j(\vec{a}) \leq u_j(\vec{b})$ and $\exists_{j \in N}\, u_j(\vec{a}) < u_j(\vec{b})$.

1.5 Nash Equilibrium

A Nash Equilibrium is a state of the game where no player prefers a different action if the current actions of the other players are fixed.

Definition An outcome a^* of a game $\langle N, (A_i), (\preceq_i) \rangle$ is a **Nash Equilibrium** if:
$\forall_{i \in N} \forall_{b_i \in A_i}\, (a^*_{-i}, b_i) \preceq (a^*_{-i}, a^*_i)$.
(a_{-i}, x) means the replacement of the value a_i with the value x.

We can look at a Nash Equilibrium as the best action that each player can play based on the given set of actions of the other players. Each player cannot profit from changing her action, and because the players are rational, this is a "steady state".

Definition Player i **Best Response** for a given set of other players actions $a_{-i} \in A_{-i}$ is the set: $BR(a_{-i}) := \{b \in A_i | \, \forall_{c \in A_i}\, (a_{-i}, c) \preceq_i (a_{-i}, b)\}$.

Under this notation, an outcome a^* is a Nash Equilibrium if $\forall_{i \in N}\, a^*_i \in BR(a^*_{-i})$.

1.6 Matrix Representation

A two player strategic game can be represented by a matrix whose rows are the possible actions of player 1 and the columns are the possible actions of player 2. Every entry in the matrix is a specific outcome and contains a vector of the payoff value of each player for that outcome.

For example, if A_1 is {r1,r2} and A_2 is {c1,c2} the matrix representation is:

	c1	c2
r1	$(w1, w2)$	$(x1, x2)$
r2	$(y1, y2)$	$(z1, z2)$

Where $u_1(r1, c2) = x1$ and $u_2(r2, c1) = y2$.

1.7 Strategic Game Examples

The following are examples of two players games with two possible actions per player. The set of deterministic Nash Equilibrium points is described in each example.

1.7.1 Battle of the Sexes

	Sports	Opera
Sports	$(2, 1)$	$(0, 0)$
Opera	$(0, 0)$	$(1, 2)$

There are two Nash Equilibrium points: (Sports, Opera) and (Opera, Sports).

1.7.2 A Coordination Game

	Attack	Retreat
Attack	$(10, 10)$	$(-10, -10)$
Retreat	$(-10, -10)$	$(0, 0)$

There are two Nash Equilibrium outcomes: (Attack, Attack) and (Retreat, Retreat). A question that raises from this game and its equilibria is how the two players can move from one Equilibrium point, (Retreat, Retreat), to the better one (Attack, Attack). Another the way to look at it is how the players can coordinate to choose the preferred equilibrium point.

1.7.3 The Prisoner's Dilemma

There is one Nash Equilibrium point: (Confess, Confess). Here, though it looks natural that the two players will cooperate, the cooperation point (Don't Confess, Don't Confess) is not a steady state since once in that state, it is more profitable for each player to move into 'Confess' action, assuming the other player will not change its action.

	Don't Confess	Confess
Don't Confess	$(-1,-1)$	$(-4,0)$
Confess	$(0,-4)$	$(-3,-3)$

1.7.4 Dove-Hawk

	Dove	Hawk
Dove	$(3,3)$	$(1,4)$
Hawk	$(4,1)$	$(0,0)$

There are two Nash Equilibrium points: (Dove, Hawk) and (Hawk, Dove).

1.7.5 Matching Pennies

	Head	Tail
Head	$(1,-1)$	$(-1,1)$
Tail	$(-1,1)$	$(1,-1)$

In this game there is no Deterministic Nash Equilibrium point. However, there is a Mixed Nash Equilibrium which is $(\frac{1}{2},\frac{1}{2}),(\frac{1}{2},\frac{1}{2})$ This is a zero sum game (the sum of the profits of each player over all possible outcomes is 0).

1.7.6 Auction

There are N players, each one wants to buy an object.

- Player i's valuation of the object is v_i, and, without loss of generality, $v_1 > v_2 > ... > v_n > 0$.
- The players simultaneously submit bids - $k_i \in [0,\infty)$. The player who submit the highest bid - k_i wins.

- In a *first price auction* the payment of the winner is the price that she bids. Her payoff is $u_i = \begin{cases} v_i - k_i, \ i = argmax\ k_i \\ 0, \ otherwise \end{cases}$.

  A Nash equilibrium point is $k_1 = v_2 + \epsilon, k_2 = v_2, ..., k_n = v_n$. In fact one can see that $k_3, \ldots, k_n$ have no influence.

In a *second price auction* the payment of the winner is the highest bid among those submitted by the players who do not win. Player i's payoff when she bids v_i is at least as high as her payoff when she submits any other bid, regardless of the other players' actions. Player 1 payoff is $v_1 - v_2$. This strategy causes the player to bid truthfully.

1.7.7 A War of Attrition

Two players are involved in a dispute over an object.

- The value of the object to player i is $v_i > 0$. Time $t \in [0, \infty)$.
- Each player chooses when to concede the object to the other player
- If the first player to concede does so at time t, her payoff $u_i = -t$, the other player obtains the object at that time and her payoff is $u_j = v_j - t$.
- If both players concede simultaneously, the object is split equally, player i receiving a payoff of $\frac{v_i}{2} - t$.

The Nash equilibrium point is when one of the players concede immediately and the other wins.

1.7.8 Location Game

- Each of n people chooses whether or not to become a political candidate, and if so which position to take.
- The distribution of favorite positions is given by the density function f on $[0, 1]$.
- A candidate attracts the votes of the citizens whose favorite positions are closer to her position.
- If k candidates choose the same position then each receives the fraction $\frac{1}{k}$ of the votes that the position attracts.
- Each person prefers to be the unique winning candidate than to tie for first place, prefers to tie the first place than to stay out of the competition, and prefers to stay out of the competition than to enter and lose.

When $n = 3$ there is no Nash equilibrium. No player wants to be in the middle, since the other players will be as close as possible to the middle player, either from the left or the right.

1.8 Mixed Strategy

Now we will expand our game and let the players' choices to be nondeterministic. Each player $i \in N$ will choose a probability distribution P_i over A_i:

1. $P = \langle P_1, ...P_N \rangle$
2. $P(\vec{a}) = \prod P_i(a_i)$
3. $u_i(P) = E_{\vec{a} \sim P}[u_i(\vec{a})]$

Note that the function u_i is linear in P_i : $U_i(P_i, \lambda\alpha_i + (1-\lambda)\beta_i) = \lambda U_i(P_{-i}, \alpha_i) + (1-\lambda)U_i(P_{-i}, \beta_i)$.

Definition $support(P_i) = \{a | P_i(a) > 0\}$

Note that the set of Nash equilibria of a strategic game is a subset of its set of mixed strategy Nash equilibria.

Lemma 1.1 *Let $G = \langle N, (A_i), (u_i) \rangle$. Then α^* is Nash equilibria of G if and only if $\forall_{i \in N} support(P_i) \subseteq BR_i(\alpha^*_{-i})$*

Proof:
$\Rightarrow$ Let α^* be a mixed strategy Nash equilibria ($\alpha^* = (P_1, ..., P_N)$). Suppose $\exists_{a \in support(P_i)}\, a \notin BR_i(\alpha^*_{-i})$. Then player i can increase her payoff by transferring probability to $a' \in BR_i(\alpha^*_{-i})$; hence α^* is not mixed strategy Nash equilibria - contradiction.
$\Leftarrow$ Let q_i be a probability distribution s.t. $u_i(Q) > u_i(P)$ in response to α^*_{-i}. Then by the linearity of u_i, $\exists_{b \in support(Q_i), c \in support(P_i)}\, u_i(\alpha^*_{-i}, b) > U_i(\alpha^*_{-i}, c)$; hence $c \notin BR_i(\alpha^*_{-i})$ - contradiction. □

1.8.1 Battle of the Sexes

As we mentioned above, this game has two deterministic Nash equilibria, (S,S) and (O,O). Suppose α^* is a stochastic Nash equilibrium:

- $\alpha^*_1(S) = 0$ or $\alpha^*_1(S) = 1 \Rightarrow$ same as the deterministic case.
- $0 < \alpha^*_1(S) < 1 \Rightarrow$ by the lemma above $2\alpha^*_2(O) = \alpha^*_2(S)$ $(\alpha^*_2(O) + \alpha^*_2(S) = 1)$ and thus $\alpha^*_2(O) = \frac{1}{3}$, $\alpha^*_2(S) = \frac{2}{3}$. Since $0 < \alpha^*_2(S) < 1$ it follows from the same result that $2\alpha^*_1(S) = \alpha^*_1(O)$ so $\alpha^*_1(S) = \frac{1}{3}$, $\alpha^*_1(O) = \frac{2}{3}$.

The mixed strategy Nash Equilibrium is $((\frac{2}{3}, \frac{1}{3}), (\frac{1}{3}, \frac{2}{3}))$.

1.9 Correlated Equilibrium

We can think of a traffic light that correlates, advises the cars what to do. The players observe an object that advises each player of her action. A player can either accept the advice or choose a different action. If the best action is to obey the advisor, the advice is a correlated equilibrium.

Definition Q is probability distribution over A. $\vec{a} \in Q$ is a Nash correlated equilibrium if $\forall z_i \in suppport(Q)\ E_Q[U_i(a_{-i}, z_i)|a_i = z_i] > E_Q[U_i(a_{-i}, x)|a_i = z_i]$

1.10 Evolutionary Equilibrium

This type of game describes an "evolution" game between different species. There are B types of species, $b, x \in B$. The payoff function is u(b,x). The game is defined as $\langle\{1, 2\}, B, (u_i)\rangle$.

The equilibrium b^* occurs when for each mutation b the payoff function satisfies $(1 - \epsilon)u(b^*, b) + \epsilon u(b, b) < (1 - \epsilon)u(b^*, b^*) + \epsilon u(b^*, b)$.

This kind of equilibrium is defined as an **evolutionarily stable strategy** since it tolerates small changes in each type.

Computational Learning Theory **Spring Semester, 2003/4**

Lecture 2: March 9

Lecturer: Yishay Mansour *Scribe: Noa Bar-Yosef, Eitan Yaffe*

2.1 Coordination Ratio

Our main goal is to compare the "cost" of *Nash equilibrium* (*NE*) to the "cost" of a global optimum of our choice. The following examples will help us get a notion of the *Coordination Ratio*:

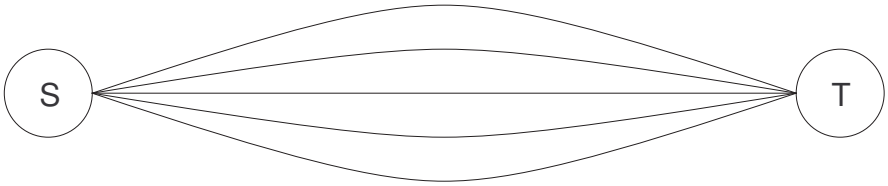


Figure 2.1: Routing on parallel lines

- Assume there is a network of parallel lines from an origin to a destination as shown in figure 2.1. Several agents want to send a particular amount of traffic along a path from the source to the destination. The more traffic on a particular line, the longer the traffic delay.
- Allocation jobs to machines as shown in figure 2.2. Each job has a different size and each machine has a different speed. The performance of each machine reduces as more jobs are allocated to it. An example for a global optimum function, in this case, would be to minimize the load on the most loaded machine.

In these scribes we will use only the terminology of the scheduling problem.

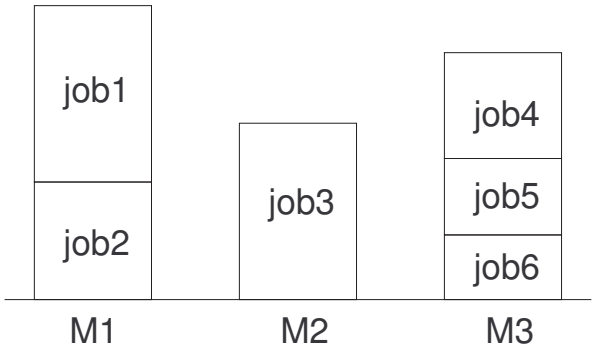


Figure 2.2: Scheduling jobs on machines

2.2 The Model

- Group of n users (or players), denoted $N = \{1, 2, ..., n\}$
- m machines: $M_1, M_2, ..., M_m$
- $\vec{s}$ speeds: $s_1, s_2, ..., s_m$ (in accordance to M_i)
- Each user i has a weight: $w_i > 0$
- ψ : mapping of users to machines:

$$\psi(i) = j$$

where i is the user and j is the machine's index. Note that NE is a special type of ψ - one which is also an equilibrium.

- The load on machine M_j will be:

$$L_j = \frac{\sum_{i:\psi(i)=j} w_j}{s_j}$$

- The cost of a configuration will be defined as the maximal load of a machine:

$$cost(\psi) = \max_j L_j$$

Our goal is to minimize the cost. The minimal cost, sometimes referred to as the *social optimum* is denoted by OPT and defined as follows:

$$OPT = \min_{\psi} cost(\psi)$$

Definition We name the ratio between the worst NE and OPT the *Coordination Ratio* and define it to be:

$$\mathbf{CR} = \frac{\max_{NE} cost(NE)}{OPT}$$

2.3 Points of equilibria

In our discussion we will attend two types of equilibria:

- Deterministic: Each user i is assigned to one machine, M_j.
- Stochastic: Each user i has a distribution p_i over $\vec{M}$. Note that the deterministic model is a special case of the stochastic model where $p_i(j) = \begin{cases} 1 & \text{if } j = j_0 \\ 0 & \text{otherwise} \end{cases}$.

When each player chooses a certain distribution, the expected load on machine j is:

$$E[L_j] = \frac{\sum_{i=1}^{n} p_i(j) * w_i}{s_j}$$

Next we define for player i the cost of choosing machine j. This function represents the point of view of player i: we define it as if he chose the machine in a deterministic manner.

$$C_i(j) = \sum_{k \neq i} \frac{p_k(j) * w_k}{s_j} + \frac{w_i}{s_j} = E[L_j] + \frac{(1 - p_i(j)) * w_j}{s_j}$$

In other words, $C_i(j)$ is the load on M_j if player i moves to machine j.

In an equilibrium player i will choose the machine with the minimal cost (and therefore he has no interest in changing to another machine). We define the cost to be:

$$Cost(i) = \min_{j} C_i(j)$$

Minimizing the cost function for player i means that $p_i(j) > 0$ only for machines that will have a minimal load after the player moves to them. For this reason, i actually shows *Best Response*. (As such, for each machine j: If $C_i(j) > Cost(i)$, then $p_i(j) = 0$. In such a case choosing M_j does not yield a Best Response).

2.4 Bounding CR

First we will show a simple bound on CR.

Claim 2.1 *For m machines, $CR \in [1, m]$.*

Proof: As any equilibrium point cannot be better than the global optimal solution, $CR >= 1$. Therefore we need only to establish the upper bound.
Let $S = \max_j s_j$. In the worst case any *Nash equilibrium* is bounded by:

$$Cost_NE \leq \frac{\sum_{i=1}^{n} w_i}{S}$$

(Otherwise, the player can move to a machine with speed S for which its load is always less than $Cost_NE$).
We also have that

$$OPT \geq \frac{\sum_{i=1}^{n} w_i}{\sum_{j=1}^{m} s_j}$$

(As if we can distribute each player's weight in an equal manner over all the machines).
Using the above bounds, we get:

$$CR = \frac{Cost_NE}{OPT} \leq \frac{\frac{\sum_{i=1}^{n} w_i}{S}}{\frac{\sum_{i=1}^{n} w_i}{\sum_{j=1}^{m} s_j}} = \frac{\sum_{j=1}^{m} s_j}{S} \leq m$$

□

Note 2.2 *The bound now for CR is linear, but in Theorem 2.9 we will show that the bound is in fact logarithmic.*

Claim 2.3 *Finding OPT for m=2, is an NP-Complete problem.*

Proof: Given that $s1 = s2$, this problem becomes identical to dividing natural numbers into two disjoint sets such that the numbers in both sets yield the same sum. This problem (called partitioning) is known to be NP-C. □

Note 2.4 *We've seen models where the optimal solution was not an equilibrium (such the 'prisoner dilema'). In this example the optimal solution is a Nash Equilibrium.*

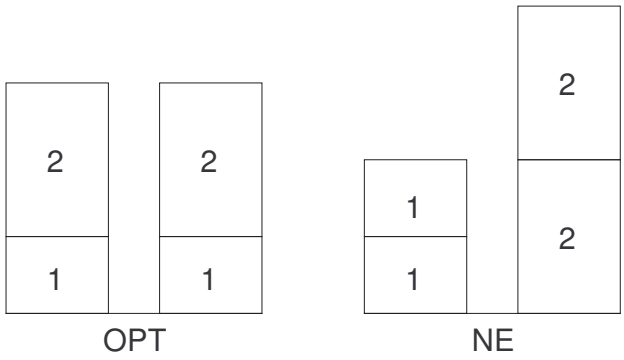


Figure 2.3: Example of $CR = \frac{4}{3}$

2.5 Two Identical Machines, Deterministic Model

As can be seen in figure 2.3, at a *Nash Equilibrium* point, the maximal load is 4. However, the maximal load of the *optimal solution* is only 3. Therefore $CR = \frac{4}{3}$.

Claim 2.5 *For 2 identical machines in the deterministic model,* $CR \geq \frac{4}{3}$.

Proof: Without loss of generality, let us assume that $L_1 > L_2$. We define $v = L_2 - L_1$.

a. If $L_2 \geq v$:
$L_1 = L_2 + v$. Therfore $Cost_NE = L_2 + v$, and OPT is at least $\frac{L_1+L_2}{2} = L_2 + \frac{v}{2}$. Hence,

$$CR = \frac{NE}{OPT} = \frac{L_2 + v}{L_2 + \frac{v}{2}} = 1 + \frac{\frac{v}{2}}{L_2 + \frac{v}{2}} \leq 1 + \frac{\frac{v}{2}}{v + \frac{v}{2}} = \frac{4}{3}.$$

b. If $L_2 < v$:
As before $L_1 = L_2 + v$. Therefore $2L_2 < L_1 < 2v$. If L_1 consists of the weight of more than one player, we will define w to be the weight of the user with the smallest weight. Since this is a *Nash Equilibrium*, $w > v$. (Otherwise the player would rather move). However, $L_1 < 2v$, hence it is not possible to have two or more players on the same machine. Because of this, we will get one player on M_1 which is the optimal solution, and $CR = 1$ accordingly.

□

2.6 Two Identical Machines, Stochastic Model

For an example we'll look at 2 identical users, for which $w_1 = w_2 = 1$, as shown in figure 2.4. Each of the players chooses a machine at random.

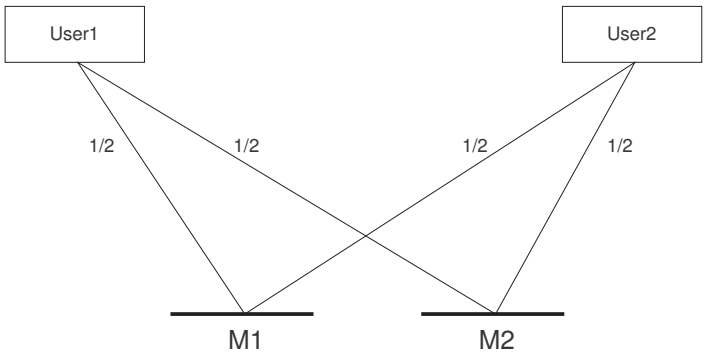


Figure 2.4: Stochastic model example

At a *Nash Equilibrium* point, with a probability of 1/2, the players will choose the same machine and with a probability of 1/2, each player will choose a different machine. Together we get $Cost_NE = 1/2 * 2 + 1/2 * 1 = 3/2$. The cost of OPT is 1 and so it follows that $CR = 3/2$.

Theorem 2.6 *For 2 identical machines in the stochastic model,* $CR \leq \frac{3}{2}$

Proof: Let $p_i(b)$ be the probability that player i chooses machine M_b. We get that

$$\bar{L_b} = E[L_b] = \sum_{i=1}^{n} (p_i(b) * w_i).$$

And the cost of player i when he chooses machine M_b becomes:

$$(E[Cost_i(b)]) = w_i + \sum_{j \neq i} (p_j(b) * w_j) = C_i(b)$$

Since we have 2 machines, $Cost(i) = \min\{C_i(1), C_i(2)\}$.
Basically, the least loaded machine, when ignoring the weight of user i, is chosen. Since each user performs according to its optimal solution, we get that in a case of an equilibrium point, if $p_i(b) > 0$ then $C_i(b) = Cost(i)$.
On the other hand, if $C_i(b) > Cost(i)$ then $p_i(b) = 0$. In other words, the player chooses her *Best Response* according to what he sees.

We now define q_i to be the probability that player i chooses the most loaded machine. We get that

$$Cost_NE = E[\max L_b] = \sum_{i=1}^{n}(q_i * w_i).$$

Furthermore, we will define the probability of a collision on a machine (both user i and user j choose the same machine) as t_{ij}.

Pay attention to the following properties:

1. In a *Nash Equilibrium* point, $\sum_{k\neq i}(t_{ik} * w_k) + w_i = Cost(i)$.
2. For m machines, $Cost(i) \leq \frac{1}{m}\sum_{k=1}^{n} w_k + \frac{m-1}{m}w_i$
   Proof:
   $$Cost(i) = \min_j C_i(j) \leq \frac{1}{m}\sum_{j=1}^{m} C_i(j)$$
   $$= \frac{1}{m}\sum_{j=1}^{m}(E[L_j] + (1 - p_i(j)) * w_i) = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{n}(p_k(j) * w_k) + \frac{m-1}{m}w_i$$
   $$= \frac{1}{m}\sum_{k=1}^{n} w_k + \frac{m-1}{m}w_i$$
   Substituting m for 2 machines, we get that
   $$Cost(i) \leq \frac{1}{2}\sum_{k=1}^{n} w_k + \frac{w_i}{2}$$
3. $q_i + q_j \leq 1 + t_{ij}$
   Proof:
   $q_i + q_j - t_{ij} \leq Pr[i \text{ and } j \text{ choose the most loaded machine}] \leq 1$.
4. $\sum_{k\neq i}(1 + t_{ik}) * w_k \leq \frac{3}{2}\sum_{k\neq i} w_k$
   Proof:
   $$\sum_{k\neq i}(1 + t_{ik}) * w_k = \sum_{k\neq i} w_k + \sum_{k\neq i} t_{ik}w_k$$
   $$= \sum_{k\neq i} w_k + Cost(i) - w_i$$
   using property 2:
   $$\leq \sum_{k\neq i} w_k + \frac{1}{2}\sum_{k} w_k + \frac{w_i}{2} - w_i$$
   $$= \frac{3}{2}\sum_{k\neq i} w_k + \frac{1}{2}w_i - \frac{1}{2}w_i$$

$$\leq \frac{3}{2}\sum_{k\neq i} w_k$$

To finish the proof of the theorem we now get:

$$Cost_NE = \sum_{k=1}^{n} q_k w_k =$$

$$\sum_k (q_i + q_k)w_k - \sum_k q_i w_k$$

$$= 2q_i w_i + \sum_{k\neq i}(q_i + q_k)w_k - q_i \sum_k w_k$$

$$\leq 2q_i w_i + \sum_{k\neq i}(1 + t_{ik})w_k - q_i \sum_k w_k$$

$$\leq 2q_i w_i + \frac{3}{2}\sum_{k\neq i} w_k - q_i \sum_k w_k$$

$$= (2q_i - \frac{3}{2})w_i + (\frac{3}{2} - q_i)\sum_k w_k$$

As previously shown, $OPT \geq \max\{\frac{1}{2}\sum_{k=1}^{n} w_k, w_i\}$.
Realize that one of the following 2 situations may occur:

1. There exists a player i such that $q_i \geq \frac{3}{4}$.
   In such a case, $(2q_i - \frac{3}{2})w_i \leq (2q_i - \frac{3}{2}) * OPT$.
   Therefore,
   $$Cost_NE \leq (\frac{3}{2} - q_i) * 2OPT + (2q_i - \frac{3}{2}) * OPT$$
   $$\leq [2q_i - \frac{3}{2} + 2(\frac{3}{2} - q_i)] * OPT$$
   $$= \frac{3}{2} * OPT$$
2. For all i, $q_i \leq \frac{3}{4}$, therefore
   $$Cost_NE = \sum_{k=1}^{n} q_k w_k \leq \frac{3}{4} * \sum_{k=1}^{n} w_k$$
   $$\leq \frac{3}{2} * OPT$$

In both cases we reach our desired result that $Cost_NE \leq \frac{3}{2} * OPT$. □

2.7 Identical machines, deterministic users

First we define some variables:

$$w_{max} = \max_i w_i \tag{2.1}$$

$$L_{max} = \max_j L_j \tag{2.2}$$

$$L_{min} = \min_j L_j \tag{2.3}$$

Claim 2.7 *In a* Nash equilibrium, $L_{max} - L_{min} \leq w_{max}$

Proof: Otherwise there would be some user j s.t. $w_j \leq w_{max}$, which could switch to the machine with load L_{min}. □

Theorem 2.8 *Given identical machines and deterministic users,* $CR \leq 2$

Proof: There are two options:

- $L_{min} \leq w_{max}$

  Then $L_{max} \leq 2w_{max}$

  But since $OPT \geq w_{max}$ we get $CR \leq \frac{L_{max}}{OPT} \leq 2$

- $L_{min} > w_{max}$

  Then $L_{max} \leq L_{min} + w_{max} \leq 2L_{min}$, which results in
  $OPT \geq \frac{1}{m}\sum_k w_k \geq L_{min}$. Therefore $CR \leq \frac{L_{max}}{OPT} \leq \frac{2L_{min}}{L_{min}} = 2$

□

2.7.1 Example of $CR \to 2$

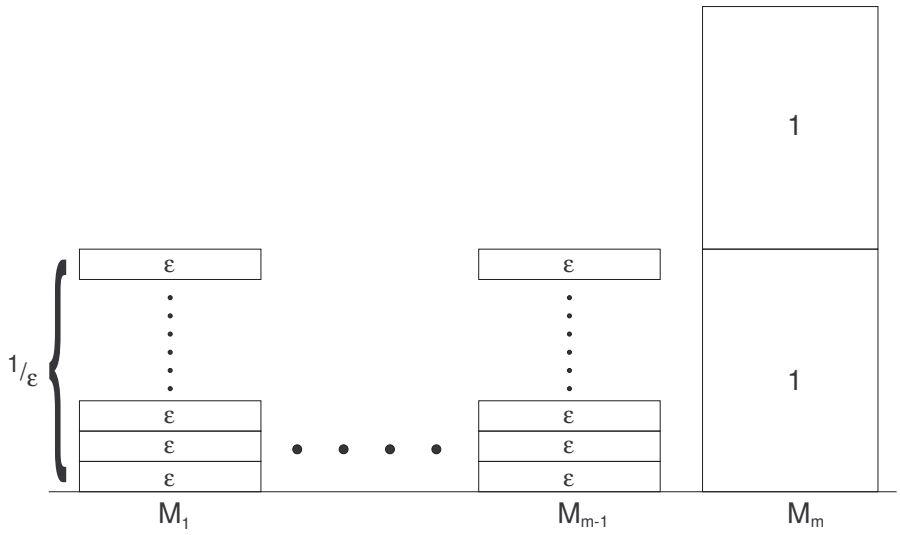


Figure 2.5: CR comes near to 2

Let's examine an example of a configuration with a CR that approaches 2. Consider m machines and $\frac{m-1}{\varepsilon}$ users with a weight of ε and 2 users with a weight of 1 as shown in figure 2.5. This is a *Nash equilibrium* with a cost of 2.

The optimal configuration is obtained by scheduling the two "heavy" users (with $w = 1$) on two separate machines and dividing the other users among the rest of the machines. In this configuration we get:

$C = OPT = 1 + \frac{1}{m} \to 1$

2.8 Identical machines, stochastic users

2.8.1 Example

Consider the following example: m machines, $n = m$ users, $w_i = 1$, $p_i(j) = \frac{1}{m}$. What is the maximal expected load?

This problem is identical to the following problem: m balls are thrown randomly into m bins; What is the expected maximum number of balls in a single bin? Let us first see what is the probability that k balls will fall into a certain bin:

$$Pr = \left(\begin{array}{c} m \\ k \end{array} \right) \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{m-k} \approx \left(\frac{c * m}{k}\right)^k \left(\frac{1}{m}\right)^k = \left(\frac{c}{k}\right)^k$$

The probability that there exists a bin with at least k balls is $1-(1-(\frac{c}{k})^k)^m$ which is constant for $k \sim \frac{\ln m}{\ln \ln m}$. Therefore the maximal load is roughly $\frac{\ln m}{\ln \ln m}$.

2.8.2 Upper bound

Using the Azuma-Hoeffding inequality we will establish a highly probable upper bound on the maximum expected load. Using theorem 2.8 from the deterministic part we know that:

$$\bar{L}_j = E[L_j] \leq 2OPT$$

We wish to prove that the probability of having a j for which $L_j \gg \bar{L}_j$ is negligible. The Azuma-Hoeffding inequality for some random variable $X = \sum x_i$, where x_i are random variables with values in the interval $[0, z]$, is:

$$P[X \geq \lambda] \leq \left(\frac{e * E[X]}{\lambda}\right)^{\frac{\lambda}{z}}$$

Let us define $\lambda = 2\alpha OPT$, $z = w_{max}$ and $x_i = \begin{cases} w_i & \text{if } p_i(j) > 0 \\ 0 & \text{otherwise} \end{cases}$

By applying the inequality we get:

$$P[L_j \geq 2\alpha OPT] \leq \left(\frac{e * E[L_j]}{2\alpha OPT}\right)^{\frac{2\alpha OPT}{w_{max}}} \leq \left(\frac{e}{\alpha}\right)^{2\alpha}$$

which results in

$$P[\exists j \ L_j \geq 2\alpha OPT] \leq m\left(\frac{e}{\alpha}\right)^{2\alpha}$$

Note that for $\alpha = \Omega(\frac{\ln m}{\ln \ln m})$ the probability is smaller than $\frac{1}{2m}$.

Theorem 2.9 *For m identical machines the worst case CR is* $O\left(\frac{\ln m}{\ln \ln m}\right)$

Proof: We shall calculate the expected cost including high loads which have a low probability, and see that their contribution is $O(1)$. For any random variable X and a natural number A we know that:

$$E[X] \leq A + \sum_{i=A}^{\infty} P[X \geq i]$$

In our case we get

$$E[\text{cost-NE}] \leq A * OPT + \sum_{\alpha=A}^{\infty} P[\text{cost-NE} \geq 2\alpha * OPT] * 2OPT$$

Therefore we define $A = 2 * c\frac{\ln m}{\ln \ln m}$ for some constant c and get

$$E[\text{cost-NE}] \leq 2 * c\frac{\ln m}{\ln \ln m} * OPT + m \sum_{\alpha} \left(\frac{e}{\alpha}\right)^{2\alpha} * OPT$$

But since $\frac{e}{\alpha} \leq \frac{1}{2m}$ we get

$$E[\text{cost-NE}] \leq 2 * c\frac{\ln m}{\ln \ln m} * OPT + O(1) * OPT$$

Resulting in

$$\text{CR} = O\left(\frac{\ln m}{\ln \ln m}\right)$$

□

2.9 Non-identical machines, deterministic users

We shall first examine a situation with a 'bad' *coordination ratio* of $\frac{\ln m}{\ln \ln m}$, then establish an upper bound.

2.9.1 Example

Let us have $k+1$ groups of machines, with N_j machines in group j. The total number of machines $m = N = \sum_{j=0}^{k} N_j$. We define the size of the groups by induction:

- $N_k = \sqrt{N}$
- $N_j = (j+1) * N_{j+1}$
- $N_0 = k! * N_k$

From the above it results that:

$$k \sim \frac{\ln N}{\ln \ln N}$$

the speed of the machines in group N_j is defined $s_j = 2^j$.

First we set up an equilibrium with a high cost. Each machine in group N_j receives j users, each with a weight of 2^j. It is easy to see that the load in group N_j is j and therefore the cost is k. Note that group N_0 received no users.

Claim 2.10 *This setup is a Nash equilibrium.*

Proof: Let us take a user in group N_j. If we attempt to move him to group N_{j-1} he will see a load of

$$(j-1)+\frac{2^j}{2^{j-1}}=j+1>j$$

On the other hand, on group N_{j+1} the load is $j+1$ even without his job and therefore he has no reason to move there. □

To achieve the optimum we simply need to move all the users of group N_j to group N_{j-1} (for $j=1...k$). Now there is a separate machine for each user and the load on all machines is $\frac{2^j}{2^{j-1}}=2$.

Corollary 2.11 *The coordination ratio is* $\sim \frac{\ln m}{\ln \ln m}$

2.9.2 Upper bound

The machines have different speeds; Without loss of generality let us assume that $s_1 \geq s_2 \cdots \geq s_m$. The cost is defined $C=\max L_j$.

For $k \geq 1$, define J_k to be the smallest index in $\{0,1,\ldots,m\}$ such that $L_{J_k+1} < k*OPT$ or, if no such index exists, $J_k = m$. We can observe the following:

- All machines up to J_k have a load of at least $k*OPT$
- The load of the machine with an index of J_k+1 is less than $k*OPT$

Let C^* be defined:

$$C^*=\lfloor\frac{C-OPT}{OPT}\rfloor$$

Our goal is to show that $C^*! < J_1$ which will result in

$$C=O\left(\frac{\log m}{\log\log m}\right)*OPT$$

We will show this using induction.

Claim 2.12 *(The induction base)* $J_{C^*} \geq 1$

Proof: By the way of contradiction, assume $J_{C^*}=0$. This implies (from the definition of J_k) that $L_1 < C^**OPT \leq C-OPT$. Let q denote the machine with the maximum expected load. Then $L_1+OPT<C=L_q$.

We observe that any user that uses q must have a weight w_i larger than s_1*OPT. Otherwise he could switch to the fastest machine, reaching a cost of $L_1+\frac{w_i}{s_1} \leq L_1+OPT<L_q$, which contradicts the stability of the *Nash equilibrium*. □

We shall divide the proof of the induction step into two claims. Let S be the group of users of the machines $M_1,\ldots,M_{J_{k+1}}$.

Claim 2.13 *An optimal strategy will not assign a user from group S to a machine $r > J_k$.*

Proof: From the definition of J_k, the users in S have a load of at least $(k+1) * OPT$. Machine $J_k + 1$ has a load of at most $k * OPT$. No user from S will want to switch to $J_k + 1$ because the minimal weight in S is $s_{J_k+1} * OPT$. Switching to machine $r > J_k + 1$ will result in a load bigger than OPT because $s_r < s_{J_k+1}$. □

Claim 2.14 *If an optimal strategy assigns users from group S to machines $1, 2, \ldots, J_k$ then $J_k \geq (k+1)J_{k+1}$*

Proof: Let $W = \sum_{i \in S} w_i$.

$$W = \sum_{j \leq J_{k+1}} s_j * E[L_j] \geq (k+1)OPT(\sum_{j \leq J_{k+1}} s_j)$$

Since an optimal strategy uses only machines $1, 2, \ldots, J_k$ we get:

$$OPT(\sum_{j \leq J_k} s_j) \geq W$$

$$\sum_{j \leq J_k} s_j \geq (k+1) * \sum_{j \leq J_{k+1}} s_j$$

Since the sequence of the speeds is non-increasing, this implies that $J_k \geq (k+1)J_{k+1}$, the induction step. □

Now we can combine the two claims above using induction to obtain:

Corollary 2.15 $C^*! < J_1$

By definition $J_1 \leq m$. Consequently $C^*! \leq m$, which implies the following:

Corollary 2.16 *(Upper bound)* $C = O(\frac{\log m}{\log \log m})$

Computational Game Theory **Spring Semester, 2003/4**

Lecture 3: Coordination Ratio of Selfish Routing

Lecturer: Yishay Mansour *Scribe: Anat Axelrod, Eran Werner*

3.1 Lecture Overview

In this lecture we consider the problem of routing traffic to optimize the performance of a congested and unregulated network. We are given a network, a rate of traffic between each pair of nodes and a latency function specifying the time needed to traverse each edge given its congestion. The goal is to route traffic while minimizing the *total latency*. In many situations, network traffic cannot be regulated, thus each user minimizes his latency by choosing among the available paths with respect to the congestion caused by other users. We will see that this "selfish" behavior does not perform as well as an optimized regulated network.
We start by exploring the characteristics of *Nash equilibrium* and *minimal latency optimal flow* to investigate the *coordination ratio*. We prove that if the latency of each edge is a linear function of its congestion, then the coordination ratio of selfish routing is at most 4/3. We also show that if the latency function is only known to be continuous and nondecreasing in the congestion, then there is no bounded coordination ratio; however, we prove that the total latency in such a network is no more than the total latency incurred by optimally routing twice as much traffic on the same network.

3.2 Introduction

We shall investigate the problem of routing traffic in a network. The problem is defined as follows: Given a rate of traffic between each pair of nodes in a network find an assignment of the traffic to paths so that the total latency is minimized. Each link in the network is associated with a latency function which is typically load-dependent, i.e. the latency increases as the link becomes more congested.
In many domains (such as the internet or road networks) it is impossible to impose regulation of traffic, and therefore we are interested in those settings where each user acts according to his own selfish interests. We assume that each user will always select the minimum latency path to its destination. In other words, we assume all users are rational and non malicious. This can actually be viewed as a noncooperative game where each user plays the best response given the state of all other users, and thus we expect the routes chosen to form a Nash equilibrium.
The network contains a numerous amount of users where each user holds only a negligible portion of the total traffic. Alternatively, we can think of a model with a finite

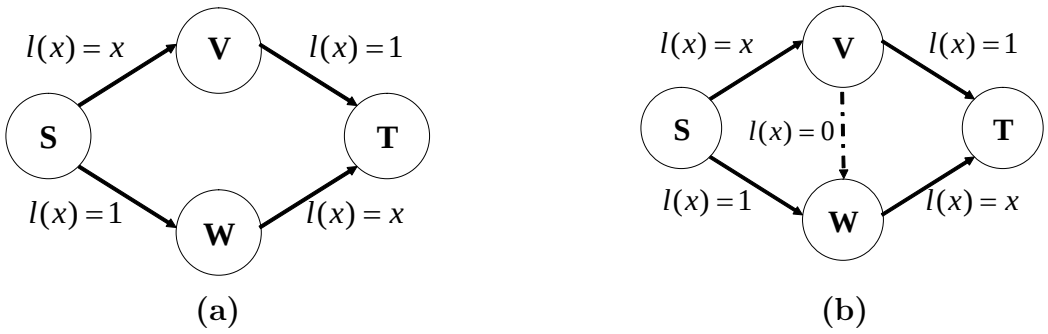


Figure 3.1:

number of users that are allowed to split their load between different paths. Our target function is to minimize the average (or total) latency suffered by all users. We will compare the overall performance under a Nash equilibrium against the theoretically optimal performance of a regulated network.

Before we continue, let's examine an example setting which has inspired much of the work in this traffic model. Consider the network in Figure 3.1(a). There are two disjoint paths from S to T. Each path follows exactly two edges. The latency functions are labelled on the edges. Suppose one unit of traffic needs to be routed from S to T. The optimal flow coincides with the Nash equilibrium such that half of the traffic takes the upper path and the other half takes the lower path. In this manner, the latency perceived by each user is $\frac{3}{2}$. In any other nonequal distribution of traffic among the two paths, there will be a difference in the total latency of the two paths and users will be motivated to reroute to the less congested path.

Note *Incidentally, we will soon realize that in any scenario in which the flow at Nash is split over more than a single path, the latency of all the chosen paths must be equal.*

Now, consider Figure 3.1(b) where a fifth edge of latency zero is added to the network. While the optimum flow has not been affected by this augmentation, Nash will only occur by routing the entire traffic on the single S $\rightarrow$ V $\rightarrow$ W $\rightarrow$ T path, hereby increasing the latency each user experiences to 2. Amazingly, adding a new zero latency link had a negative effect for all agents. This counter-intuitive impact is known as *Braess's paradox*.

Anecdote 1 *Two live and well known examples of Braess's paradox occurred when 42nd street was closed in New York City and instead of the predicted traffic gridlock, traffic flow actually improved. In the second case, traffic flow worsened when a new road was constructed in Stuttgart, Germany, and only improved after the road was torn up.*

3.2.1 The Model - Formal Definition

- We consider a directed graph $\boldsymbol{G = (V,E)}$ with $\boldsymbol{k}$ pairs $\boldsymbol{(s_i, t_i)}$ of source and destination vertices.
- r_i - The amount of flow required between s_i and t_i.
- P_i - The set of simple paths connecting the pair (s_i, t_i). $\mathcal{P} = \bigcup_i P_i$.
- Flow $\boldsymbol{f}$ - A function that maps a path to a positive real number. Each path P is associated with a flow f_P.
- f_e - The flow on edge e defined for a fixed flow function. $f_e = \sum_{P:e\in P} f_P$.
- A flow f is said to be *feasible* if $\forall i,\ \sum_{P\in P_i} f_P = r_i$.
- Each edge $e \in E$ is given a load-dependent latency function denoted $\ell_e(\cdot)$. We restrict our discussion to nonnegative, differentiable and nondecreasing latency functions.
- (G, r, ℓ) - A triple which defines an *instance* of the routing problem.
- The latency of a path ℓ_P is defined as the sum of latencies of all edges in the path. $\ell_P(f) = \sum_{e\in P} \ell_e(f_e)$.
- $\boldsymbol{C(f)}$ - The total latency, also defined as the *cost* of a flow f. $C(f) = \sum_{P\in\mathcal{P}} \ell_P(f) f_P$. Alternatively, we can accumulate over the edges to get $C(f) = \sum_{e\in E} \ell_e(f_e) f_e$.

3.3 Characterizations of Nash & OPT Flows

3.3.1 Flows at Nash Equilibrium

Lemma 3.3.1 *A feasible flow f for instance (G, r, ℓ) is at Nash equilibrium iff for every $i \in \{1, ..., k\}$ and $P_1, P_2 \in P_i$ with $f_{P_1} > 0$, $\ell_{P_1}(f) \le \ell_{P_2}(f)$.*

From the lemma it follows that flow at Nash equilibrium will be routed only on best response paths. Consequently, all paths assigned with a positive flow between (s_i, t_i) have equal latency denoted by $L_i(f)$.

Corollary 3.1 *If f is a flow at a Nash equilibrium for instance (G, r, ℓ) then $C(f) = \sum_{i=1}^{k} L_i(f) r_i$.*

3.3.2 Optimal (Minimum Total Latency) Flows

Recall that a cost of a flow f is expressed by $C(f) = \sum_{e \in E} \ell_e(f_e) f_e$. We seek to minimize this function for finding an optimal solution.

Observation 3.2 *Finding the minimum latency feasible flow is merely a case of the following non-linear program:*

$$\min \sum_{e \in E} c_e(f_e)$$

subject to:

$$\textit{(NLP)} \qquad \begin{array}{ll} \sum_{P \in P_i} f_P = r_i & \forall i \in \{1, ...k\} \\ f_e = \sum_{P \in \mathcal{P}: e \in P} f_P & \forall e \in E \\ f_P \geq 0 & \forall P \in \mathcal{P} \end{array}$$

where in our problem we assign $c_e(f_e) = \ell_e(f_e) f_e$.

Note *For simplicity the above formulation of (NLP) is given with an exponential number of variables (there can be an exponential number of paths). This formulation can be easily modified with decision variables only on edges giving a polynomial number of variables and constraints.*

In our case we assume that for each edge $e \in E$ the function $c_e(f_e) = \ell_e(f_e) f_e$ is a convex function and therefore, our target function *C(f)* is also convex. This is a special case of convex programming. We wish to optimize (minimize) a convex function *F(x)* where x belongs to a convex domain.
Recall the following properties of convex sets and functions:

1. If f is strictly convex then the solution is unique.
2. If f is convex then the solution set U is convex.
3. If y is not optimal ($\exists x : F(x) < F(y)$) then y is not a local minimum. Consequently, any local minimum is also the global minimum.

Lemma 3.3.2 *The flow f is optimal for the convex program of the form (NLP) iff* $\forall i \in \{1, ..., k\}$ *and* $P_1, P_2 \in P_i$ *with* $f_{P_1} > 0$, $c'_{P_1}(f) \leq c'_{P_2}(f)$.

Notice the striking similarity between the characterization of optimal solutions (Lemma 3.3.2) and Nash equilibrium (Lemma 3.3.1). In fact, an optimal flow can be interpreted as a Nash equilibrium with respect to a different edge latency functions.
Let $x\ell_e(x)$ be a convex function for all $e \in E$. Define $\ell^*_e(f_e) = (\ell_e(f_e) f_e)'$.

Corollary 3.3 *A feasible flow f is an optimal flow for* (G, r, ℓ) *iff it is at Nash equilibrium for the instance* (G, r, ℓ^*).

Proof. f is OPT for $\ell \Leftrightarrow c'_{P_1}(f) \leq c'_{P_2}(f) \Leftrightarrow \ell^*_{P_1}(f) \leq \ell^*_{P_2}(f) \Leftrightarrow f$ is Nash for ℓ^* $(\forall i \; \forall P_1, P_2 \in P_i)$. □

3.3.3 Existence of Flows at Nash Equilibrium

We exploit the similarity between the characterizations of Nash and OPT flows to establish that a Nash equilibrium indeed exists and its cost is unique.
For the outline of the proof we define an edge cost function $h_e(x) = \int_0^x \ell_e(t)dt$.
By definition $(h_e(f_e))' = \frac{d}{dx}h_e(f_e) = \ell_e(f_e)$ thus h_e is differentiable with non decreasing derivative ℓ_e and therefore convex. Next, we consider the following convex program:

$$\min \sum_{e\in E} h_e(f_e)$$

subject to:

$$\text{(NLP2)} \qquad \begin{array}{ll} \sum_{P\in P_i} f_P = r_i & \forall i \in \{1,...,k\} \\ f_e = \sum_{P\in\mathcal{P}:e\in P} f_P & \forall e \in E \\ f_P \geq 0 & \forall P \in \mathcal{P} \end{array}$$

Observation 3.4 *The optimal solution for (NLP2) is Nash for the modified instance where $\ell_e(x) = h'_e(x)$.*

Proof. The proof follows directly from Lemma 3.3.1 and Lemma 3.3.2 □

Since Nash is an optimal solution for a different convex setting we conclude that:

- Nash equilibrium exists.
- The cost at Nash equilibrium is unique.

3.3.4 Bounding the Coordination ratio

The relationship between Nash and OPT characterizations provide a general method for bounding the coordination ratio $\rho = \frac{C(f)}{C(f^*)} = \frac{Nash}{OPT}$.

Theorem 3.5 *For an instance (G, r, ℓ), if there exists a constant $\alpha \geq 1$ s.t.*

$$\forall e \in E\ \forall x \in \mathbb{R}^+ \quad x\ell_e(x) \leq \alpha \int_0^x \ell_e(t)dt$$

then $\rho(G, r, \ell) \leq \alpha$.

Proof.

$$\begin{aligned} C(f) &= \sum_{e\in E} \ell_e(f_e)f_e \\ &\leq \alpha \sum_{e\in E} \int_0^{fe} \ell_e(t)dt \\ &\leq \alpha \sum_{e\in E} \int_0^{f_e^*} \ell_e(t)dt \\ &\leq \alpha \sum_{e\in E} \ell_e(f_e^*)f_e^* \\ &= \alpha \cdot C(f^*) \end{aligned}$$

The first inequality follows from the hypothesis, the second follows from the fact that Nash flow f is OPT for the function $\sum_{e\in E}\int_0^x \ell_e(t)dt$ and the final inequality follows from the assumption that the latency functions ℓ_e are nondecreasing.
□

Corollary 3.6 *If every latency function ℓ_e has the form $\ell_e(x) = \sum_{i=0}^{d} a_{e,i}x^i$ (meaning latency is a polynomial function of order d) then $\rho(G, r, \ell) \leq d + 1$.*

Note *From the corollary, an immediate coordination ratio of 2 is established for linear latency functions. Later, we will show a tighter bound of $\frac{4}{3}$.*

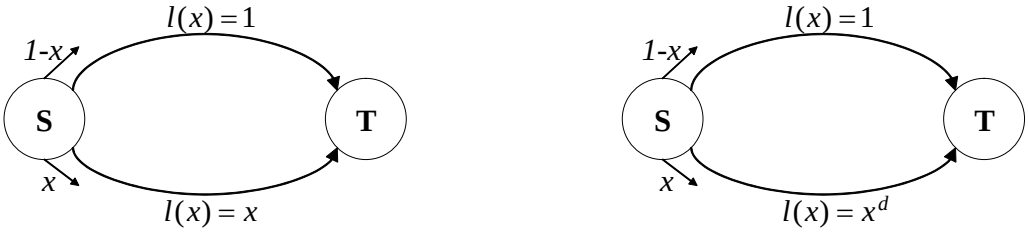


Figure 3.2:

Figure 3.2(a) shows an example for which Nash flow will only traverse in the lower path while OPT will divide the flow equally among the two paths. The target function is $1(1 - x) + x \cdot x$ and it reaches minimum with value $\frac{3}{4}$ when x = $\frac{1}{2}$, giving a coordination ratio of $\frac{4}{3}$ for this example. Combining the example with the tighter upper bound to be shown, we demonstrate a tight bound of $\frac{4}{3}$ for linear latency functions.

In Figure 3.2(b) the flow at Nash will continue to use only the lower path but OPT will reach minimum for the cost function $x \cdot x^d + (1 - x) \cdot 1$ at $x = (d + 1)^{-\frac{1}{d}}$, giving a total latency 1-$\frac{d}{d+1}(d + 1)^{-1/d}$ which approaches 0 as d $\to \infty$. So, $\lim_{d\to\infty} \rho = \infty$ meaning, ρ cannot be bounded from above in some cases when nonlinear latency functions are allowed.

3.4 A Bicriteria Bound for latency functions

We now examine an interesting *bicriteria* result. We show that the cost of a flow at Nash equilibrium can be bounded by the cost of an optimal flow feasible for twice the amount of traffic.

Theorem 3.7 *If f is a flow at Nash equilibrium for instance (G, r, ℓ) and f^* is a feasible flow for instance $(G, 2r, \ell)$ (same network but with twice the required rate), then C(f) $\leq C(f^*)$.*

Proof. Let $L_i(f)$ be the latency of a $s_i - t_i$ flow path, so that $C(f) = \sum_i L_i(f) r_i$. We define a new latency function:

$$\bar{\ell}_e(x) = \begin{cases} \ell_e(f_e) & \text{if } x \leq f_e \\ \ell_e(x) & \text{if } x \geq f_e \end{cases}$$

This latency function will allow us to approximate the original latencies as well as to lower bound the cost of any feasible flow.

- Step 1: Let's compare the cost of f^* under the new latency function $\bar{\ell}$ with respect to the original cost $C(f^*)$.
  From the construction of $\bar{\ell}_e(x)$ we get:

  $$\begin{array}{ll} \bar{\ell}_e(x) - \ell_e(x) = 0 & \text{for } x \geq f_e \\ \bar{\ell}_e(x) - \ell_e(x) \leq \ell_e(f_e) & \text{for } x \leq f_e \end{array}$$

  So, for all x we get $x[\bar{\ell}_e(x) - \ell_e(x)] \leq \ell_e(f_e) f_e$.

  The difference between the new cost under $\bar{\ell}_e$ and the original cost under ℓ is:

  $$\begin{aligned} \textstyle\sum_e \bar{\ell}_e(f_e^*) f_e^* - C(f^*) &= \textstyle\sum_{e \in E} f_e^* (\bar{\ell}_e(f_e^*) - \ell_e(f_e^*)) \\ &\leq \textstyle\sum_{e \in E} \ell_e(f_e) f_e \\ &= C(f). \end{aligned}$$

  The cost of OPT with the latency function $\bar{\ell}$ increased by at most the cost of Nash (an additive $C(f)$ factor).

- Step 2: Denote z_0 the zero flow in G. For the pair $s_i - t_i$ we can observe that by construction, $\forall P \in P_i \;\; \bar{\ell}_P(z_0) \geq \ell_P(f) \geq L_i(f)$.
  Hence, since $\bar{\ell}_e$ is nondecreasing for each edge e, $\forall P \in P_i \;\; \bar{\ell}_P(f^*) \geq \bar{\ell}_P(z_0) \geq \ell_P(f) \geq L_i(f)$, revealing that the cost of f^* with respect to $\bar{\ell}$ can be bounded as follows:

  $$\begin{aligned} \textstyle\sum_P \bar{\ell}_P(f^*) f_P^* &\geq \textstyle\sum_i \sum_{P \in P_i} L_i(f) f_P^* \\ &= \textstyle\sum_i 2 L_i(f) r_i = 2C(f). \end{aligned}$$

Combining the results from the previous two steps finishes the proof of the theorem:

$$\begin{aligned} C(f^*) &\geq \textstyle\sum_P \bar{\ell}_P(f^*) f_P^* - C(f) \\ &\geq 2C(f) - C(f) = C(f). \end{aligned}$$

□

3.5 A Tight Bound for Linear Latency Functions

Finally, we consider a scenario where all edge latency functions are linear $\ell_e(x) = a_e x + b_e$, for constants $a_e, b_e \geq 0$. A fairly natural example for such a model is a network employing a congestion control protocol such as TCP. We have already seen in Figure 3.2(a) an example where the coordination ratio was $\frac{4}{3}$. We have also established an upper bound of 2 according to Corollary 3.6. We shall now show that the $\frac{4}{3}$ ratio is also a tight upper bound. Prior to this result, we examine two simple cases:

1. $\ell_e(x) = b$
2. $\ell_e(x) = a_e x$.

For both these cases we will show that OPT=Nash.

- Case 1 is obvious since the latency on each path is constant, so both OPT and Nash will route all the flow to the paths with minimal latency.
- Case 2:
  - Using Lemma 3.3.1, a flow f is at Nash equilibrium *iff* for each source-sink pair i and $P, P' \in P_i$ with $f_P > 0$ then $\ell_P(f) = \sum_{e \in P} \ell_e(f_e) = \sum_{e \in P} a_e f_e \leq \sum_{e' \in P'} a_{e'} f_{e'} = \ell_{P'}(f)$.
  - Using Lemma 3.3.2, a flow f^* is an optimal flow *iff* for each source-sink pair i and $P, P' \in P_i$ with $f^*_P > 0$ then $C'_P(f^*) = \sum_{e \in P} C'_e(f^*_e) = \sum_{e \in P} ((a_e f^*_e) f^*_e)' = \sum_{e \in P} 2a_e f^*_e \leq \sum_{e' \in P'} 2a_{e'} {f^*}'_e = C'_{P'}(f^*)$.

Corollary 3.8 *For the latency functions $\ell_e(x) = a_e(x)$ f is at Nash equilibrium iff f is an optimal flow.*

Observation 3.9 *In the example shown in Figure 3.2(a) we showed that even a simple combination of the two sets of functions is enough to demonstrate that OPT $\neq$ Nash.*

Theorem 3.10 *Let f be a flow at Nash equilibrium and f^* an optimal flow . If the latency functions are all of the form $\ell_e(x) = a_e x + b_e$ then $\rho \leq \frac{4}{3}$.*

Proof. We define a new latency function $\bar{\ell}_e$,

$$\bar{\ell}_e(x) = (\ell_e(f_e)) \cdot x = \ell^f_e \cdot x$$

Under this definition of $\bar{\ell}_e$, $OPT \equiv Nash$ (by Corollary 3.8).
Hence, f is at Nash equilibrium with respect to $\bar{\ell} \Leftrightarrow$ for every feasible flow x where $C^f(\cdot)$ is the cost with respect to $\bar{\ell}$, $C^f(f) \leq C^f(x)$.

$$\begin{aligned} C^f(x) &= \textstyle\sum_e (a_e f_e + b_e) \cdot x_e \\ &\leq \textstyle\sum_e (a_e x_e + b_e) x_e + \frac{1}{4} \sum_e a_e f_e^2 \\ &\leq C(x) + \tfrac{1}{4} C(f) \end{aligned}$$

The first inequality is justified by the following algebraic steps:

$$\begin{aligned} & (a_e f_e + b_e) x_e && \leq (a_e x_e + b_e) x_e + \tfrac{a_e f_e^2}{4} \\ \Leftrightarrow \quad & a_e f_e x_e && \leq a_e x_e x_e + \tfrac{a_e f_e^2}{4} \\ \Leftrightarrow \quad & f_e x_e && \leq x_e^2 + \tfrac{f_e^2}{4} \\ \Leftrightarrow \quad & 0 && \leq x_e^2 - f_e x_e + \tfrac{f_e^2}{4} = (x_e - \tfrac{f_e}{2})^2 \end{aligned}$$

Since f brings $C^f(\cdot)$ to minimum,

$$\begin{aligned} C(f) = C^f(f) &\leq C^f(x) \\ &\leq C(x) + \tfrac{1}{4} C(f) \end{aligned}$$

or,

$$\frac{3}{4} C(f) \leq C(x).$$

As this is true for all x, let's plug-in $x = f^*$:

$$C(f) \leq \frac{4}{3} C(f^*).$$

□

3.6 FIN

All good things must come to an end.

Lecture 4: 2-Player Zero Sum Games

Lecturer: Yishay Mansour *Scribe: Yair Halevi, Daniel Deutch*

4.1 2-Player Zero Sum Games

In this lecture we will discuss 2-player zero sum games. Such games are completely competitive, where whatever one player wins, the other must lose. Examples of such games include chess, checkers, backgammon, etc. We will show that in such games:

- An equilibrium always exists;
- All equilibrium points yield the same payoff for all players;
- The set of equilibrium points is actually the cartesian product of independent sets of equilibrium strategies per player.

We will also show applications of this theory.

Definition Let G be the game defined by $\langle N, (A_i), (u_i)\rangle$ where N is the number of players, A_i is the set of possible pure strategies for player i, and u_i is the payoff function for player i. Let A be the cartesian product $A = \prod_{i=1}^{n} A_i$. Then G is a *zero sum game* if and only if:

$$\forall \vec{a} \in A, \ \sum_{i=1}^{n} u_i(\vec{a}) = 0 \tag{4.1}$$

In other words, a *zero sum game* is a game in which, for any outcome (any combination of pure strategies, one per player), the sum of payoffs for all players is zero.

We naturally extend the definition of u_i to any probability distribution $\vec{p}$ over A by $u_i(\vec{p}) = E_{\vec{a} \sim \vec{p}}(u_i(\vec{a}))$. The following is immediate due to the linearity of the expectation and the zero sum constraint:

Corollary 4.1 *Let G be a zero sum game, and Δ the set of probability distributions over A. Then*

$$\forall \vec{p} \in \Delta, \ \sum_{i=1}^{n} u_i(\vec{p}) = 0 \tag{4.2}$$

Specifically, this will also hold for any probability distribution that is the product of N independent distributions, one per player, which applies to our normal mixed strategies game.

A *2-player zero sum game* is a zero sum game with $N = 2$. In this case, 4.1 may be written as

$$\forall a_1 \in A_1, a_2 \in A_2, \quad u_1(a_1, a_2) = -u_2(a_1, a_2) \tag{4.3}$$

Such a game is completely competitive. There is no motivation for cooperation between the players.

A two person zero sum game may also be described by a single function $\pi : A_1 \times A_2 \to R$ describing the payoff value for player I, or the loss value for player II. The goal of player I is to maximize π, while the goal of player II is to minimize π. We say that $\pi(i, j)$ is the *value of the game for strategies i and j* or simply the *payoff for i and j*.

Given a certain ordering of the pure strategies of both players, we can also represent a finite 2-player zero sum game using a real matrix $A_{m \times n}$ (the *payoff matrix*), where m is the number of pure strategies for player I and n is the number of pure strategies for player II. The element a_{ij} in the ith row and jth column of A is the payoff (for player I) assuming player I chooses his ith strategy and player II chooses his jth strategy.

4.2 Nash Equilibria

The Nash equilibria of a 2-player zero sum game have several interesting properties. First, they all exhibit the same value. Second, they are *interchangeable*, meaning that given 2 Nash equilibrium points, it is possible to replace a strategy for one of the players in the first point by the strategy of the same player in the second point and obtain another Nash equilibrium. Formally:

Theorem 4.2 *Let G be a 2-player zero sum game defined by $\langle (A_1, A_2), \pi \rangle$. Let (τ_1, τ_2) and (σ_1, σ_2) be two Nash equilibria for G. Then*

1. *Both (σ_1, τ_2) and (τ_1, σ_2) are Nash equilibria of G.*
2. $\pi(\tau_1, \tau_2) = \pi(\tau_1, \sigma_2) = \pi(\sigma_1, \tau_2) = \pi(\sigma_1, \sigma_2)$

Proof: (σ_1, σ_2) is a Nash equilibrium. Therefore, for the first player (who plays to maximize π), we have

$$\pi(\sigma_1, \sigma_2) \geq \pi(\tau_1, \sigma_2)$$

However, (τ_1, τ_2) is a Nash equilibrium as well. Therefore, for the second player (who plays to minimize π) we have

$$\pi(\tau_1, \sigma_2) \geq \pi(\tau_1, \tau_2)$$

Combining these two inequalities we get

$$\pi(\sigma_1, \sigma_2) \geq \pi(\tau_1, \sigma_2) \geq \pi(\tau_1, \tau_2)$$

Similarly,

$$\pi(\sigma_1, \sigma_2) \leq \pi(\sigma_1, \tau_2) \leq \pi(\tau_1, \tau_2)$$

From the last two inequalities we obtain

$$\pi(\sigma_1, \sigma_2) = \pi(\tau_1, \tau_2) = \pi(\sigma_1, \tau_2) = \pi(\tau_1, \sigma_2)$$

Which proves part 2 of the theorem. To prove part 1 we observe that because (σ_1, σ_2) is a Nash equilibrium for player I,

$$\forall \alpha_1' \in A_1, \quad \pi(\alpha_1', \sigma_2) \leq \pi(\sigma_1, \sigma_2) = \pi(\tau_1, \sigma_2)$$

Where the right-hand equation is due to part 2 of the theorem which has already been proven. Similarly, because (τ_1, τ_2) is a Nash equilibrium for player II,

$$\forall \alpha_2' \in A_2, \quad \pi(\tau_1, \alpha_2') \geq \pi(\tau_1, \tau_2) = \pi(\tau_1, \sigma_2)$$

Which means that (τ_1, σ_2) is a Nash equilibrium as well. The proof is similar for (σ_1, τ_2). □

Theorem 4.2 holds with the same proof for both the deterministic and the nondeterministic case.

We define the *equilibrium strategies* of a player as the set of all strategies played by the player in any equilibrium point. For player I, this is given by

$$\{\sigma_1 \in A_1 \mid \exists \sigma_2 \in A_2, \ \ (\sigma_1, \sigma_2) \text{ is an eq. pt. } \}$$

Corollary 4.3 *The set of Nash equilibrium points of a 2-player zero sum game is the cartesian product of the equilibrium strategies of each player.*

When a 2-player zero sum game is represented as a matrix A, a deterministic Nash equilibrium for the game is a saddle point of A, or a pair of strategies i, j so that

$$a_{ij} = \max_k a_{kj}$$

$$a_{ij} = \min_l a_{il}$$

Such an equilibrium does not necessarily exist.

4.3 Payoff Bounds

For a deterministic game, player I can guarantee a payoff lower bound by choosing a pure strategy for which the minimal payoff is maximized. This assumes player II is able to know player I's choice and will play the worst possible strategy for player I (note that in a 2-player zero sum game this is also player II's best response to player I's chosen strategy).

We denote this "gain-floor" by V_I':

$$V_I' = \max_i \min_j a_{ij}$$

Similarly, player II can guarantee a loss upper bound by choosing the pure strategy for which the maximal payoff is minimal. We denote this "loss-ceiling" by V_{II}':

$$V_{II}' = \min_j \max_i a_{ij}$$

Lemma 4.4 *For any function $F : X \times Y \to R$, for which all the relevant minima and maxima exist:*

1. $\max_{x\in X} \min_{y\in Y} F(x,y) \leq \min_{y\in Y} \max_{x\in X} F(x,y)$
2. *Equality holds iff:*

$$\exists x_0 \in X, y_0 \in Y, \quad F(x_0,y_0) = \min_{y\in Y} F(x_0,y) = \max_{x\in X} F(x,y_0)$$

Proof: The proof of this lemma is trivial and is not shown here. □

Applying Lemma 4.4 to our case proves the intuitive fact that player I's gain-floor cannot be greater than player II's loss-ceiling,

$$V_I' \leq V_{II}'$$

and that equality holds iff we have a saddle point and thus an equilibrium.

4.4 Mixed Strategies

For a finite 2-player zero sum game denoted as a matrix $A_{m\times n}$, we denote a mixed strategy for a player I (II) by a stochastic vector of length m (n), where the ith element in the vector is the probability for the ith pure strategy of this player (using the same order used to generate the payoff matrix).

Vectors in this text are always row vectors. We will typically use x for player I mixed strategies, and y for player II mixed strategies. We shall denote by Δ_d the set of stochastic vectors in R^d.

For a 2-player zero sum game given by matrix $A_{m\times n}$, and given mixed strategies x for player I and y for player II, the expected payoff is given by

$$A(x,y) = \sum_{i=1}^{m}\sum_{j=1}^{n} x_i a_{ij} y_j = xAy^T \tag{4.4}$$

Once again, if player I chose strategy x, the minimum gain (which is also player II's best response loss) is

$$v_{II}(x) = \min_{y\in\Delta_n} xAy^T \tag{4.5}$$

Assuming player II knows what player I has played before selecting a strategy. The minimum exists because Δ_n is compact and xAy^T is continuous in y. It is easily shown that this minimum must be reachable in at least one pure strategy of player II.

Lemma 4.5

$$\forall x \in \Delta_m,\ \ v_{II}(x) = \min_{y\in\Delta_n} xAy^T = \min_{1\le j\le n} xAe_j^T$$

Proof: The proof is trivial given the fact that xAy^T is a stochastic combination of xAe_j^T, so xAy^T can never be less than all of xAe_j^T, and on the other hand, e_j is also in Δ_n, so $v_{II}(x) \le xAe_j^T$.

□

Therefore we can write 4.5 as

$$v_{II}(x) = \min_{1\le j\le n} xAe_j^T = \min_{1\le j\le n}\sum_{i=1}^{m} x_i a_{ij} \tag{4.6}$$

Which means that player I can guarantee the following lower bound on his payoff (gain-floor)

$$V_I = \max_{x\in\Delta_m}\min_{y\in\Delta_n} xAy^T = \max_{x\in\Delta_m}\min_{1\le j\le n} xAe_j^T = \max_{x\in\Delta_m}\min_{1\le j\le n}\sum_{i=1}^{m} x_i a_{ij} \tag{4.7}$$

Such a mixed strategy x that maximizes $v_{II}(x)$ is a *maximin* strategy for player I. Once again, this maximum exists due to compactness and continuity.

We define $v_I(y)$ in a similar fashion as player I's most harmful response (to player II) to strategy y of player II (this is also player I's best response to y). Then, player II can guarantee the following upper bound on his loss (loss-ceiling)

$$V_{II} = \min_{y\in\Delta_n}\max_{x\in\Delta_m} xAy^T = \min_{y\in\Delta_n}\max_{1\le i\le m} e_i Ay^T = \min_{y\in\Delta_n}\max_{1\le i\le m}\sum_{j=1}^{n} y_j a_{ij} \tag{4.8}$$

Such a mixed strategy y that maximizes $v_I(y)$ is a *minimax* strategy for player II.

V_I and V_{II} are called the *values* of the game for players I and II, respectively.

4.5 The Minimax Theorem

Applying Lemma 4.4 to the maximin and minimax values of the game we obtain

$$V_I \leq V_{II} \tag{4.9}$$

In fact, we will show the following fundamental property of 2-player zero sum games

Theorem 4.6 *(The Minimax Theorem)*

$$V_I = V_{II}$$

We start by proving two lemmas.

Lemma 4.7 *(Supporting Hyperplane Theorem) Let $B \subseteq R^d$ be a closed convex set and $\vec{x} \notin B$ then $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ and α_{d+1} exist such that*

$$\vec{\alpha} \cdot \vec{x} = \sum_{i=1}^{d} \alpha_i x_i = \alpha_{d+1} \tag{4.10}$$

$$\forall y \in B, \quad \vec{\alpha} \cdot \vec{y} = \sum_{i=1}^{d} \alpha_i y_i > \alpha_{d+1} \tag{4.11}$$

In other words, given a convex closed set B and a point outside the set $\vec{x}$, the lemma claims that we can pass a hyperplane through $\vec{x}$ such that B lies entirely on one side of the hyperplane. This lemma and it's proof are schematically shown in figure 4.1.

Proof: Let $\vec{z} \in B$ be the point in B nearest to $\vec{x}$. Such a point exists because B is closed, and the distance function is both continuous and bounded from below by 0. We define

$$\begin{aligned} \vec{\alpha} &= \vec{z} - \vec{x} \\ \alpha_{d+1} &= \vec{\alpha} \cdot \vec{x} \end{aligned}$$

4.10 holds immediately. We shall prove 4.11. Note that $\vec{\alpha} \neq 0$ because $\vec{z} \in B$ and $\vec{x} \notin B$. Thus,

$$\vec{\alpha} \cdot \vec{z} - \alpha_{d+1} = \vec{\alpha} \cdot \vec{z} - \vec{\alpha} \cdot \vec{x} = \vec{\alpha} \cdot (\vec{z} - \vec{x}) = \vec{\alpha} \cdot \vec{\alpha} = |\vec{\alpha}|^2 > 0$$

Therefore,

$$\vec{\alpha} \cdot \vec{z} > \alpha_{d+1}$$

Now, assume that there exists $\vec{y} \in B$ such that

$$\vec{\alpha} \cdot \vec{y} \leq \alpha_{d+1}$$

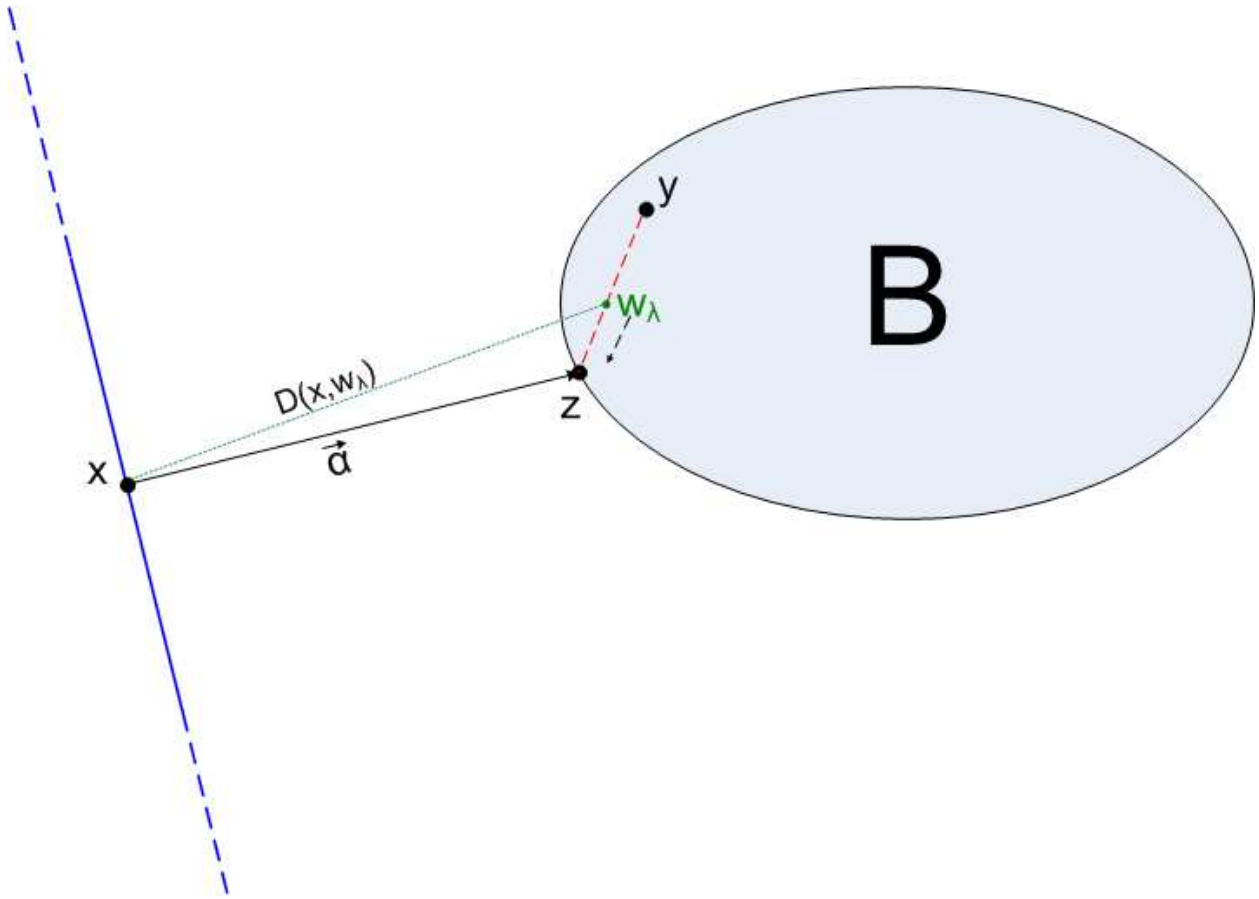


Figure 4.1: Supporting Hyperplane

As B is convex, for any $0 \leq \lambda \leq 1$,

$$\vec{w}_\lambda = \lambda \vec{y} + (1-\lambda)\,\vec{z} \in B$$

The square of the distance between $\vec{x}$ and $\vec{w}_\lambda$ is given by

$$D^2\left(\vec{x}, \vec{w}_\lambda\right) = \left|\vec{x} - \lambda\vec{y} - (1-\lambda)\,\vec{z}\right|^2 = \sum_{i=1}^{d} \left(x_i - \lambda y_i - (1-\lambda)\,z_i\right)^2$$

Deriving by λ we obtain

$$\begin{aligned} \frac{\partial D^2}{\partial \lambda} &= 2\left(\vec{x} - \lambda\vec{y} - (1-\lambda)\,\vec{z}\right)\cdot\left(\vec{z} - \vec{y}\right) \\ &= 2\left(\vec{z} - \vec{x}\right)\cdot\vec{y} - 2\left(\vec{z} - \vec{x}\right)\cdot\vec{z} + 2\lambda\left(\vec{z} - \vec{y}\right)^2 \\ &= 2\vec{\alpha}\cdot\vec{y} - 2\vec{\alpha}\cdot\vec{z} + 2\lambda\left(\vec{z} - \vec{y}\right)^2 \end{aligned}$$

Evaluating for $\lambda = 0$ we get

$$\frac{\partial D^2}{\partial \lambda} = 2\vec{\alpha} \cdot \vec{y} - 2\vec{\alpha} \cdot \vec{z}$$

But according to our assumption the first term $\vec{\alpha} \cdot \vec{y} \leq \alpha_{d+1}$ and we have shown that the second term $\vec{\alpha} \cdot \vec{z} > \alpha_{d+1}$, and therefore

$$\left.\frac{\partial D^2}{\partial \lambda}\right|_{\lambda=0} < 0$$

Hence, for λ close enough to 0 we must have

$$D^2(\vec{x}, \vec{w}_\lambda) < D^2(\vec{x}, \vec{z})$$

But $\vec{z}$ was chosen to minimize the distance to $\vec{x}$, so we have a contradiction. Therefore for all $\vec{y} \in B$, 4.11 must hold.

□

Lemma 4.8 *(Theorem of the Alternative for Matrices). Let $A = (a_{ij})$ be an $m \times n$ real matrix. Let $\{\vec{a}_i\}_{i=1}^m = (a_{i1}, a_{i2}, \ldots, a_{in})$ be the rows of the matrix. Then one of the following must hold:*

1. *The point $\vec{0}$ in R^n is in the convex hull of the $m+n$ points $\{\vec{a}_i\}_{i=1}^m \bigcup \{\vec{e}_i\}_{i=1}^n$ where $\vec{e}_i$ is the ith elementary vector in R^n.*
2. *There exists a stochastic vector $\vec{x} = (x_1, \ldots, x_n) \in R^n$ satisfying*

$$\begin{array}{lll} & \sum_{j=1}^n x_j & = 1 \\ \forall 1 \leq j \leq n, & x_j & > 0 \\ \forall 1 \leq i \leq m, & \vec{a}_i \cdot \vec{x} = \sum_{j=1}^n a_{ij} x_j & > 0 \end{array}$$

Proof: Suppose 1 does not hold. Denote the convex hull mentioned in 1 by C. If we apply Lemma 4.7, there exist $\vec{\alpha} \in R^n$ and α_{n+1} such that

$$\vec{\alpha} \cdot \vec{0} = \alpha_{n+1}$$

(which means that $\alpha_{n+1} = 0$, of course) and

$$\forall \vec{y} \in C, \quad \vec{\alpha} \cdot \vec{y} > 0$$

In particular, this will hold if $\vec{y}$ is any of the vectors $\vec{a}_i$ or $\vec{e}_i$. Thus

$$\begin{array}{rll} \vec{a}_i \cdot \vec{\alpha} & > 0 & \text{for all } 1 \leq i \leq m, \\ \alpha_j & > 0 & \text{for all } 1 \leq j \leq n. \end{array}$$

Since $\forall 1 \leq j \leq n,\ \ \alpha_j > 0$ we have $\sum_{j=1}^{n} \alpha_j > 0$, so we can scale by the sum and define

$$x_j = \alpha_j \Big/ \sum_{j=1}^{n} \alpha_j$$

Therefore

$$\begin{array}{lcl} & \sum_{j=1}^{n} x_j & = 1 \\ \forall 1 \leq j \leq n, & x_j & > 0 \\ \forall 1 \leq i \leq m, & \vec{a}_i \cdot \vec{x} = \sum_{j=1}^{n} a_{ij} x_j & > 0 \end{array}$$

□

Proof of the Minimax Theorem: Let $A_{m \times n}$ be a payoff matrix for a 2-player zero sum game. Applying Lemma 4.8 to A^T, either 1 or 2 must hold. If 1 holds, then $\vec{0}$ is in the convex hull of the columns of A and the elementary vectors in R^m. Thus, there exist $s_1, \ldots, s_{n+m}$ such that

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} s_j + s_{n+i} &= 0 \quad \forall 1 \leq i \leq m \\ s_i &\geq 0 \quad \forall 1 \leq i \leq n+m \\ \sum_{i=1}^{n+m} s_i &= 1 \end{aligned}$$

Now, it is impossible for all of $s_1, \ldots, s_n$ to be equal to 0, because the first equation would mean that all s_i are 0, and then equation 3 cannot hold (in other words, the vector $\vec{0}$ cannot be a convex combination of $\vec{e}_i$ alone, because they are linearly independent). Therefore at least one of $s_1, \ldots, s_n$ is positive, and $\sum_{k=1}^{n} s_k > 0$. We can therefore define a mixed strategy y for player II by

$$\forall 1 \leq j \leq n, \qquad y_j = s_j \Big/ \sum_{k=1}^{n} s_k$$

And we have:

$$\begin{aligned} y_j &\geq 0 \quad \forall 1 \leq j \leq n \\ \sum_{j=1}^{n} y_j &= 1 \\ \sum_{j=1}^{n} a_{ij} y_j = -s_{n+i} \Big/ \sum_{k=1}^{n} s_k &\leq 0 \quad \forall 1 \leq i \leq m \end{aligned}$$

Therefore

$$v_I(y) = \max_{1 \le i \le m} e_i A y^T \le 0$$

Thus $V_{II} \le 0$.

If 2 holds, then we have a stochastic vector $x \in R^m$, which we will view as a mixed strategy for player I, that satisfies

$$\forall 1 \le j \le n, \sum_{i=1}^{m} a_{ij} x_i > 0$$

so $v_{II}(x) = \min_{1 \le j \le n} x A e_j^T > 0$ and therefore $V_I > 0$. Because one of the two must hold, we see that it is impossible to have $V_I \le 0 < V_{II}$.

Now, for any real M, let us look at the 2-player zero sum game defined by the payoff matrix $B = (b_{ij})$ where

$$b_{ij} = a_{ij} - M$$

For any x, y,

$$x B y^T = x A y^T - M$$

Hence,

$$\begin{aligned} V_I(B) &= V_I(A) - M \\ V_{II}(B) &= V_{II}(A) - M \end{aligned}$$

And since it is impossible that $V_I(B) \le 0 < V_{II}(B)$, it is also impossible that

$$V_I(A) \le M < V_{II}(A)$$

But this is true for any real M, thus it is impossible that

$$V_I < V_{II}$$

And we have shown that $V_I \le V_{II}$, therefore

$$V_I = V_{II}$$

□

4.6 Results

We have shown that in a 2-player zero sum game the gain-ceiling for player I is equal to the loss-floor for player II. We denote this value simply by V and call it the *value of the game.*

Part 2 of Lemma 4.4 tells us that $V_I = V_{II}$ means that we have a Nash equilibrium point. It is easy to see that the payoff in this equilibrium is exactly the value of the game. Theorem 4.2 tells us that all Nash equilibria will have this value, and that the set of all Nash equilibria is actually a cartesian product of the equilibrium strategies of each player.

A strategy x for player I satisfying

$$\forall 1 \leq j \leq n, \quad \sum_{i=1}^{m} x_i a_{ij} \geq V \tag{4.12}$$

is optimal for player I in the sense that this strategy guarantees a payoff of V against every strategy of player II, and there is no strategy that guarantees a higher payoff against every strategy of player II. Similarly, a strategy y for player II satisfying

$$\forall 1 \leq i \leq m, \quad \sum_{j=1}^{n} y_j a_{ij} \leq V \tag{4.13}$$

is optimal for player II. It is clear that

$$xAy^T = V$$

otherwise one of 4.12 or 4.13 will not hold. It is easy to see that (x, y) is a Nash equilibrium. Also, any Nash equilibrium must satisfy 4.12 and 4.13.

To summarize

Theorem 4.9 *Let G be a 2-player zero sum game. Then*

1. *The gain-floor for player I and loss-ceiling for player II are equal (the value of the game, V).*
2. *There is at least one Nash equilibirium.*
3. *The set of equilibrium points for the game is the cartesian product of the sets of equilibrium strategies for each player.*
4. *The value of the game in all equilibrium points is V*
5. *The set of equilibrium strategies for each player is equal to the set of optimal strategies for the player*
6. *The set of optimal strategies for player I is the solution of the following linear program in variables $x_1, \ldots, x_m, V$*

$$\begin{array}{lcl} \forall 1 \leq j \leq n, & \sum_{i=1}^{m} x_i a_{ij} - V & \geq 0 \\ \forall 1 \leq i \leq m & x_i & \geq 0 \\ & \sum_{i=1}^{m} x_i & = 1 \\ & \textit{Maximize target function} & V \end{array}$$

and the dual program gives the optimal strategies for player II.

The problem of finding Nash equilibria for a 2-player zero sum game is therefore in NP $\bigcap$ co-NP, and is solvable in polynomial time by linear programming.

4.7 Application of Zero Sum Games in Computer Science

4.7.1 Deterministic vs. Random Algorithms

In this example $\Omega = \{A_i\}$ is a finite set of deterministic algorithms that can take as input any element of the finite input set $\Lambda = \{x_j\}$. We will denote by Δ_S the set of probability distributions over the set S, for any set S.

Definition **Time**$(\mathbf{A}, \mathbf{x})$ is the time complexity (measured, as usual in complexity, in the means of number of commands) of running the deterministic algorithm A with the input x. Also denoted $T(A, x)$.

Definition A **Random Algorithm** is a probability distribution over the deterministic algorithms, $\vec{p} \in \Delta_\Omega$. We denote the probability for algorithm A_i by p_i.

Definition **RTime**$(\vec{\mathbf{p}}, \mathbf{x})$ is time complexity of the random algorithm defined by distribution $\vec{p}$ for fixed input x. It is defined as the expected deterministic time complexity for the fixed input x:

$$RTime(\vec{p}, x) = \sum_i p_i \cdot T(A_i, x)$$

Definition **AvgTime**$(\mathbf{A}, \vec{\mathbf{q}})$ is the time complexity of deterministic algorithm A given distribution $\vec{q}$ over inputs. This is in essence an average-case complexity analysis for A. It is defined as the expected time complexity for the deterministic algorithm A with input distributed according to $\vec{q}$:

$$AvgTime(A, \vec{q}) = \sum_j q_j \cdot T(A, x_j)$$

Complexity Analysis

Corollary 4.10 *Deterministic worst-case time complexity is* $\min_i \max_j T(A_i, x_j)$.

Proof: The complexity of the problem is the minimum complexity over all relevant algorithms (Ω). We must choose the deterministic algorithm before knowing the input. Thus, the complexity of deterministic algorithm A_i is analyzed for the worst input, which yields complexity $\max_j T(A, x_j)$, and then the complexity of the problem is the complexity of the best algorithm, which results in complexity $\min_i \max_j T(A_i, x_j)$. □

Corollary 4.11 *Non-deterministic worst-case time complexity is* $\max_j \min_i T(A_i, x_j)$

Proof: For non-deterministic algorithms we can guess the best deterministic algorithm given the input. Thus, for input x_j, the complexity is $\min_i T(A_i, x_j)$. We now analyze for the worst case input, which yields complexity $\max_j \min_i T(A_i, x_j)$. □

Corollary 4.12 *Random worst-case time complexity is*

$$\min_{\vec{p} \in \Delta_\Omega} \max_j RTime(\vec{p}, x_j)$$

Theorem 4.13 ***(Yao's Lemma)*** *For any distribution $\vec{p}$ on the algorithms and $\vec{q}$ on inputs*

$$\max_j E_{i \sim \vec{p}} [T(A_i, x_j)] \geq \min_i E_{j \sim \vec{q}} [T(A_i, x_j)]$$

Proof: We can view the complexity analysis as a 2-player zero sum game in the following way. The max. player pure strategies are the possible inputs, Λ. The min. player pure strategies are the deterministic algorithms Ω. The payoff is the time complexity $T(A_i, x_j)$.

Given such a game, we can see that

$$\min_{\vec{p} \in \Delta_\Omega} \max_{\vec{q} \in \Delta_\Lambda} E_{i \sim \vec{p}, j \sim \vec{q}} [T(A_i, x_j)] = \max_{\vec{q} \in \Delta_\Lambda} \min_{\vec{p} \in \Delta_\Omega} E_{i \sim \vec{p}, j \sim \vec{q}} [T(A_i, x_j)] \tag{4.14}$$

As in the previous game analysis, it is easily shown that the internal maximum and minimum are obtained in deterministic points:

$$\max_{\vec{q} \in \Delta_\Lambda} E_{i \sim \vec{p}, j \sim \vec{q}} [T(A_i, x_j)] = \max_j E_{i \sim \vec{p}} [T(A_i, x_j)] \tag{4.15}$$

$$\min_{\vec{p} \in \Delta_\Omega} E_{i \sim \vec{p}, j \sim \vec{q}} [T(A_i, x_j)] = \min_i E_{j \sim \vec{q}} [T(A_i, x_j)] \tag{4.16}$$

Using only the $\geq$ part of 4.14, and substituting using 4.15 and 4.16 we obtain

$$\min_{\vec{p} \in \Delta_\Omega} \max_j E_{i \sim \vec{p}} [T(A_i, x_j)] \geq \max_{\vec{q} \in \Delta_\Lambda} \min_i E_{j \sim \vec{q}} [T(A_i, x_j)] \tag{4.17}$$

Hence for any $\vec{p} \in \Delta_\Omega$

$$\max_j E_{i \sim \vec{p}} [T(A_i, x_j)] \geq \max_{\vec{q} \in \Delta_\Lambda} \min_i E_{j \sim \vec{q}} [T(A_i, x_j)]$$

Thus for any $\vec{p} \in \Delta_\Omega$ and $\vec{q} \in \Delta_\Lambda$

$$\max_j E_{i\sim\vec{p}}[T(A_i, x_j)] \geq \min_i E_{j\sim\vec{q}}[T(A_i, x_j)]$$

□

Note that Yao's Lemma is actually a result of the weaker inequality established in Lemma 4.4.

Corollary 4.14 *In order to prove a lower bound for the worst-case complexity of any random algorithm for a given problem, it is sufficient to prove a lower bound for any deterministic algorithm on some distribution of the input.*

Proof: Using

$$\begin{aligned} E_{i\sim\vec{p}}[T(A_i, x_j)] &= RTime(\vec{p}, x_j) \\ E_{j\sim\vec{q}}[T(A_i, x_j)] &= AvgTime(A_i, \vec{q}) \end{aligned}$$

we can write Yao's Lemma as

$$\max_j RTime(\vec{p}, x_j) \geq \min_i AvgTime(A_i, \vec{q})$$

so given a lower bound B on the complexity of any deterministic algorithm on some input distribution $\vec{q}$, we obtain

$$B \leq \min_i AvgTime(A_i, \vec{q}) \leq \max_j RTime(\vec{p}, x_j)$$

So B is a lower bound on the worst-case complexity of any random algorithm. □

Example - Sorting a List of Numbers

We wish to bound the complexity of a random algorithm for sorting n numbers (comparison based sort). We can describe any deterministic comparison based sort algorithm as a decision tree, where each internal node corresponds to a comparison the algorithm performs, with 2 possible outcomes (we assume all elements are different). For a specific input, the exectuion of the algorithm corresponds to a path from the root to a leaf. It is impossible for 2 different permutations to result in the same path. The running time for the algorithm over an input is the length of the path.

Therefore, the decision tree must have at least $n!$ leaves. Thus the depth of the tree is at least $log(n!) = O(nlogn)$ nodes. The number of leaves whose depth is not greater than l is $\leq 2^{l+1}$.

Thus, for any deterministic algorithm A, at least one half of the permutations are in depth greater than l, where $l+1 = log(n!/2)$ (since then the number of leaves whose depth is less than l is $\leq 2^{log(n!/2)} = n!/2$). $l+1 = log(n!/2) \Longrightarrow l = log(n!) - 2 = O(nlogn)$.

We shall choose a uniform distribution $\vec{q}$ over the possible inputs (all permutations of n numbers), and fix a deterministic algorithm A. The running time of A over this distribution is simply the average of the depths of the leaves for all possible inputs. But at least $n!/2$ inputs are of depth at least $log(n!) - 1$, so the average running time will be at least

$$\frac{\frac{n!}{2} \cdot (log(n!) - 1)}{n!} = \Omega(nlogn)$$

And using Yao's lemma, the complexity of any random algorithm is also $\Omega(nlogn)$.

4.7.2 Weak vs. Strong Learning

Given a weak learner for a binary classification problem we will show that strong learning is possible.

The model: f is the target function, H a function family.

$$f : X \longrightarrow \{0,1\}$$

$$\forall h \in H, \quad h : X \longrightarrow \{0,1\}$$

X is finite, and as a consequence H is finite ($|H| \leq 2^{|X|}$)

The WL (weak learning) assumption: For every distribution D on X there exists $h \in H$ and $\epsilon > 0$ such that

$$[Pr_D[h(x) = f(x)] \geq 1/2 + \epsilon \tag{4.18}$$

Question: can f be approximated by functions in H?

We represent the problem as a 2-player zero sum game as follows. The max. player pure strategies are the inputs X. The min. player pure strategies are the functions H. The payoff is an error indicator:

$$M(h,x) = \left\{ \begin{array}{lll} 0 & \text{if} & f(x) = h(x) \text{ (no error)} \\ 1 & \text{if} & f(x) \neq h(x) \text{ (error)} \end{array} \right\}$$

Note that $M(h,x) = |(f(x) - h(x)|$. The max. player is trying to select a distribution over X that will maximize the expected error, while the min. player is trying to select a distribution over H that will minimize it.

The WL proposition implies that for each D there exists an $\epsilon > 0$ and $h \in H$ such that

$$1/2 - \epsilon \geq Pr_D[h(x) \neq f(x)] = Pr_D[M(h,x) = 1] = E_{x \sim D} D[M(h,x)]$$

Thus there exists an $\epsilon > 0$ so that

$$\min_h E_{x\sim D}[M(h,x)] \leq 1/2 - \epsilon$$

Which means that

$$V_{X\ Player} \leq 1/2 - \epsilon$$

Since in a zero sum game, the values of both players are equal, we conclude that

$$V_{H\ Player} < 1/2 - \epsilon$$

hence

$$\min_q \max_x E_{h\sim q}[M(h,x)] \leq 1/2 - \epsilon$$

therefore there exists a distribution q (the one in which the minimum is obtained) such that

$$\max_x E_{h\sim q}[M(h,x)] \leq 1/2 - \epsilon$$

thus

$$\forall x \in X, \quad E_{h\sim q}[M(h,x)] \leq 1/2 - \epsilon$$

In other words, for this q, and all $x \in X$,

$$1/2 - \epsilon \geq \sum_{h\in H} q(h)M(h,x) = \sum_{h\in H} q(h)\,|f(x) - h(x)|$$

We define an approximation $G(x) = \sum_{h\in H} q(h) \cdot h(x)$. Now, for all $x \in X$,

$$|f(x) - G(x)| = \left|\sum_{h\in H} q(h)[f(x) - h(x)]\right| \leq \sum_{h\in H} q(h)\,|f(x) - h(x)| < 1/2$$

So $G(x)$ is a strong approximation for $f(x)$ (by rounding $G(x)$ to 0 or 1 we obtain $f(x)$ for all $x \in X$.

4.7.3 Playing a Zero-Sum Game

Let A be the game matrix. In each time t:

- The rows player chooses a distribution p_t.
- The columns player chooses a distribution q_t (the player may or may not know p_t before choosing q_t).
- The rows player loses $p_t A q_t^T$. His goal is to minimize his total loss, $\sum_t p_t A q_t^T$
- After playing time t, the rows player is also given the vector Aq_t^T, so he knows what his loss would have been in time t for any strategy he would have played.

KL distance

Definition KL(Kullback-Leibler) distance is defined as follows

$$KL(P_1\|P_2) = \sum_x P_1(x) ln\left(\frac{P_1(x)}{P_2(x)}\right)$$

Characteristics: $KL(P_1\|P_1) = 0$, and for every $P1 \neq P2$, $KL(P1\|P2) > 0$.

An algorithm

We attach a weight for each action, at each time. The weights are updated as follows:

$$w_{t+1}(i) = w_t(i)\beta^{L_t(i)}$$

where $\beta \in (0,1)$ is a parameter,

$$L_t(i) = \left(Aq_t^T\right)_i = e_i A q_t^T$$

is the loss caused by strategy i at time t, and

$$w_1(i) = 1$$

At time t the algorithm chooses a distribution p_t such that

$$p_t(i) = w_t(i) \Big/ \sum_{j=1}^{N} w_t(j)$$

Denoting

$$z_t = \sum_{j=1}^{N} w_t(j)$$

Theorem 4.15 *For any game matrix A and for any $q_1, \cdots q_t$ the $p_1, \cdots, p_t$ that the algorithm chooses satisfy:*

$$Loss = \sum_t p_t A q_t^T \leq \min_p \left[a_\beta \sum_t p\left(Aq_t^T\right) + c_\beta KL(p\|p_1)\right]$$

where

$$\begin{aligned} a_\beta &= \frac{ln(1/\beta)}{1-\beta} \\ c_\beta &= \frac{1}{1-\beta} \end{aligned}$$

Lemma 4.16 *For every iteration t and for every $\tilde{p}$*

$$KL(\tilde{p}\|p_{t+1}) - KL(\tilde{p}\|p_t) \leq ln\left(\frac{1}{\beta}\right)\ \tilde{p}Aq_t^T + ln\left(1-(1-\beta)p_tAq_t^T\right)$$

Proof:

$$KL(\tilde{p}\|p_{t+1}) - KL(\tilde{p}\|p_t) = \sum_i \tilde{p}(i)ln\left(\frac{p_t(i)}{p_{t+1}(i)}\right) = \sum_i \tilde{p}(i)ln\left(\frac{z_{t+1}}{\beta^{L_t(i)}z_t}\right) =$$

$$ln\left(\frac{z_{t+1}}{z_t}\right) + \sum_i \tilde{p}(i)ln\left(\frac{1}{\beta}\right)^{L_t(i)} = ln\left(\frac{z_{t+1}}{z_t}\right) + ln\left(\frac{1}{\beta}\right)\ \sum_i \tilde{p}(i)L_t(i) =$$

$$ln\left(\frac{z_{t+1}}{z_t}\right) + ln\left(\frac{1}{\beta}\right)\ \tilde{p}Aq_t^T \tag{4.19}$$

But

$$ln\frac{z_{t+1}}{z_t} = ln\frac{\sum_i w_{t+1}(i)}{\sum_i w_t(i)} = ln\sum_i \frac{w_t(i)}{\sum_j w_t(j)}\cdot\beta^{L_t(i)} = ln\sum_i p_t(i)\beta^{L_t(i)}$$

Using $\beta^x \leq 1-(1-\beta)x$ for $x\in[0,1]$ we obtain

$$ln\frac{z_{t+1}}{z_t} \leq ln\left(\sum_i p_t(i)\left(1-(1-\beta)L_t(i)\right)\right)\ = ln\left(1-(1-\beta)\textstyle\sum_{i=1}^N p_t(i)L_t(i)\right) =$$
$$ln\left(1-(1-\beta)p_tAq_t^T\right) \tag{4.20}$$

Combining 4.19 and 4.20

$$KL(\tilde{p}\|p_{t+1}) - KL(\tilde{p}\|p_t) \leq ln\left(1-(1-\beta)p_tAq_t^T\right) + ln\left(\frac{1}{\beta}\right)\ \tilde{p}Aq_t^T \tag{4.21}$$

□

Proof of the Theorem:
Let $\tilde{p}$ be some distribution over the rows. Since $ln(1-x)\leq -x$ for $x<1$ we have,

$$ln\left(1-(1-\beta)p_tAq_t^T\right) \leq -(1-\beta)p_tAq_t^T$$

Hence from Lemma 4.16,

$$KL(\tilde{p}\|p_{t+1}) - KL(\tilde{p}\|p_t) \leq -(1-\beta)p_tAq_t^T + ln\left(\frac{1}{\beta}\right)\ \tilde{p}Aq_t^T \tag{4.22}$$

Summing 4.22 for all $t = 1, \ldots, T$ yields a telescope sum, resulting in

$$KL(\tilde{p}\|p_{T+1}) - KL(\tilde{p}\|p_1) \leq -(1-\beta)\sum_{t=1}^{T} p_t A q_t^T + ln\left(\frac{1}{\beta}\right)\ \sum_{t=1}^{T} \tilde{p} A q_t^T$$

But since $KL(\tilde{p}\|p_{T+1}) \geq 0$ we obtain

$$(1-\beta)\sum_{t=1}^{T} p_t A q_t^T \leq KL(\tilde{p}\|p_1) + ln\left(\frac{1}{\beta}\right)\ \sum_{t=1}^{T} \tilde{p} A q_t^T$$

Thus

$$Loss = \sum_{t=1}^{T} p_t A q_t^T \leq c_\beta KL(\tilde{p}\|p_1) + a_\beta \sum_{t=1}^{T} \tilde{p} A q_t^T$$

For any distribution $\tilde{p}$, which proves the theorem.

□

We can now use the theorem to bound the average loss per step for our algorithm. Substituting p_1 to be uniform distribution ($w_1(i) = 1$) means $KL(\tilde{p}\|p_1) \leq ln(N)$, because

$$KL(\tilde{p}\|p_1) = \sum_x \tilde{p}(x) ln\left(\frac{\tilde{p}(x)}{1/N}\right) = \sum_x \tilde{p}(x) ln(\tilde{p}(x)\cdot N) \leq \sum_x \tilde{p}(x) ln(N) \leq ln(N)$$

So now we have

$$\sum_{t=1}^{T} p_t A q_t^T \leq c_\beta ln N + a_\beta \sum_{t=1}^{T} \tilde{p} A q_t^T \tag{4.23}$$

Choosing $\beta = \frac{1}{1+\sqrt{\frac{2ln(n)}{T}}} \sim 1 - \Theta\left(\sqrt{ln\frac{N}{T}}\right)$ we get

$$\frac{1}{T} Loss = \frac{1}{T}\sum_{t=1}^{T} p_t A q_t^T \leq \min_p \frac{1}{T}\sum_{t=1}^{T} p A q_t^T + \Delta$$

Where $\Delta = \sqrt{\frac{2lnN}{T}} + \frac{lnN}{T}$. This is achieved by substituting our choice of β in 4.23 and using the approximation $ln(1/\beta) \leq (1-\beta^2)/(2\beta)$ for $\beta \in (0, 1]$.

The meaning of this is that the difference between average loss per step to that of the optimal fixed strategy is bounded by a value that be made arbitrarily small for large T.

Computational Game Theory **Spring Semester, 2003/04**

Lecture 5: March 30

Lecturer: Yishay Mansour *Scribe: Nataly Sharkov,Itamar Nabriski*

5.1 Introduction

In this lecture we explore issues concerning the computability of Nash Equilibrium in any general game. First we prove any game has at least one Nash Equilibrium using Brouwer's Fixed Point Theorem (The proof is presented at section 5.6). Then we discuss how to compute a Nash Equilibrium of a game and a Nash Equilibrium approximation called ϵ-Nash. We conclude by proving that the computation of most information concerning Nash Equilibrium is NP-hard.

5.2 Proof of existence of Stochastic Nash Equilibrium in any game

5.2.1 Proof Outline

We prove that any game has a Nash Equilibrium, though not necessarily a deterministic one. Using Brouwer's Fixed Point Theorem, we prove that in any game if we map every game state to another state, such that at least one player is better off, then there is some state which is mapped into itself (a fixed point). In other words, this game state cannot be improved by any player by changing his strategy, and thus is, by definition, a Nash Equilibrium.

5.2.2 Notations

- There are n players: $N = \{1, ..., n\}$
- Each player can choose from a set of m pure strategies, thus the possible Strategies for each Player i are: $A_i = \{a_{i1}, ..a_{im}\}$
- u_i - utility function of player i
- p_i - distribution over A_i
- $\wp = \prod p_i$ - product of players' distributions
- $(q, \wp_{-i})$ - taking $\wp$ with p_i replaced by q
- $u_i(\wp) = E_{\vec{u}\sim\wp}[u_i(\vec{a})]$ - expected utility of player i

- Δ_i - all possible distributions for player i (infinite set - since each distribution contains m values in the range $[0, 1]$)
- $\Lambda = \prod \Delta_i$ - product of all players' possible distributions

5.2.3 Definitions

- Nash Equilibrium - $\wp^*$ is a Nash Equilibrium if:

$$\wp^* \in \Lambda$$

$$\forall i \in N : q_i \in \Delta_i$$

$$u_i(q_i, \wp^*_{-i}) \leq u_i(\wp^*)$$

- Revenue of player i from deterministic action $a_{i,j}$:

$$Revenue_{i,j}(\wp) = u_i(a_{i,j}, \wp_{-i})$$

- Profit of player i from deterministic action $a_{i,j}$:

$$Profit_{i,j}(\wp) = (Revenue_{i,j}(\wp) - u_i(\wp))$$

- Gain of player i from deterministic action $a_{i,j}$:

$$Gain_{i,j} = max(Profit_{i,j}(\wp), 0)$$

  $Gain_{i,j}$ is non zero only if player i has positive profit from playing strategy $a_{i,j}$ instead of his strategy in $\wp$. Thus in Nash Equilibrium ($\wp = \wp^*$) all $Gain_{i,j}$'s are zero.

- We define a mapping:

$$y : \Lambda \to \Lambda$$

$$y_{i,j}(\wp) = \frac{p_{i,j} + Gain_{i,j}}{1 + \sum_{k=1}^{m} Gain_{i,k}}$$

  - $y_{i,*}$ is a distribution over A_i (where $*$ means for any j)
  - $y_{i,j}$ is continuous (since $Revenue_{i,j}$ is continuous $\rightarrow$ $Profit_{i,j}$ is continuous $\rightarrow$ $Gain_{i,j}$ is continuous $\rightarrow$ $y_{i,j}$ is continuos)
  - Notice that the denominator of the expression on the right is there in order to normalize the expression to be a value in $\wp$.

5.2.4 Proof

By Brouwer's Theorem (see section 5.6) for a continuous function $f : S \to S$ such that S is a convex and compact set, there exists $s \in S$ such that $s = f(s)$. s is called a fixed point of f. Thus in our case, for y, there exists $\wp^*$:

$$y_{i,j}(\wp^*) = p_{i,j} = \frac{p_{i,j} + Gain_{i,j}}{1 + \sum_{k=1}^{m} Gain_{i,k}}$$

We argue that this point is a Nash Equilibrium.

- If $\sum_{k=1}^{m} Gain_{i,k} = 0$ then we are done since all $Gain_{i,k}$ are zero and thus there is no room for improvement which is by definition a Nash Equilibrium.
- Otherwise $\sum_{k=1}^{m} Gain_{i,k} > 0$:

$$p_{i,j}(1 + \sum_{k=1}^{m} Gain_{i,k}) = p_{i,j} + Gain_{i,j} \Longrightarrow$$

$$p_{i,j} \sum_{k=1}^{m} Gain_{i,k} = Gain_{i,j}$$

This implies one of two cases:

1. $$p_{i,j} = 0 \Longleftrightarrow Gain_{i,j} = 0$$

   Then we are in a Nash Equilibrium and the fixed point is 0.

2. $$p_{i,j} > 0 \Longleftrightarrow Gain_{i,j} > 0$$

   This implies that for every strategy that has a positive probability $a_{i,k}$, playing it purely will net a higher utility. $p_{i,j}$ is a distribution, thus, if we take $p_{i,j}$ over all strategies we will net a higher utility. But this implies that we can improve $p_{i,j}$ by selecting $p_{i,j}$, which is a contradiction

5.3 Computing of Nash equilibria

5.3.1 General sum games

We begin from the example of the general sum game. Let A,B be payoff matrices of player 1 and 2 accordingly.

$$A = \begin{pmatrix} 1 & 5 & 7 & 3 \\ 2 & 3 & 4 & 3 \end{pmatrix}$$

$$B = \begin{pmatrix} 2 & 3 & 1 & 5 \\ 4 & 1 & 6 & 0 \end{pmatrix}$$

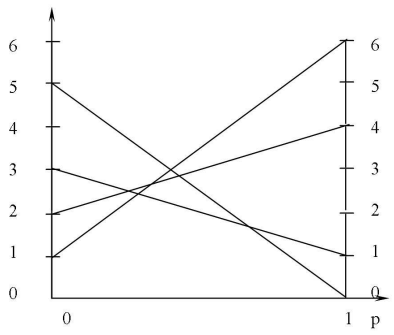


Figure 5.1: The viewpoint of player 2.

Player 1 has only two strategies: (x, 1-x).
Player 2 has vector of strategies. Let number all strategies of player 2. $j \in [1, 4]$ - the index of player 2's strategies. The utility of playing strategy j is $b_{1j}x + b_{2j}(1 - x)$.

$p \in [0, 1]$ is the probability that player 1 plays bottom strategy. (1-p) that he plays top strategy. We can receive Nash equilibrium in points those player 2 is indifferent between strategies 3 and 4. Let see all cases.

- The side (3,3). For this side the player 1 is indifferent, since for all values p lengthwise the side that is equilibrium.

- Extreme points. Let see the case of extreme points p=0, p=1. In these points Equilibrium can be received if the action p is better than $\overline{x}$ - reverse action.

- Pass points. Check the pass points from one to other side. There are two such points. The first one is (3,3)-(1,2) pass point. In this point player 2 prefer other strategy. The second one is (1,2)-(7,4). There is Nash equilibrium in this point. Let compute it.

$$1 \cdot p + 7 \cdot (1 - p) = 2p + 4(1 - p)$$

$$(7 - 4)(1 - p) = (2 - 1)p$$

$$3 = \frac{7 - 4}{2 - 1} = \frac{p}{1 - p}$$

$$p = \frac{3}{4}$$

$$1 - p = \frac{1}{4}$$

In the same way we can compute the case when player 1 is indifferent between top and bottom strategies. For player 2 those are strategies number three and fourth.

5.3.2 An algorithm that uses the support vectors

Consider a general two-players game
A- payoff matrix of player 1.
B- payoff matrix of player 2.
As stipulated by Nash's theorem a Nash Equilibrium (p,q) is exists.
Suppose we know the support sets. p has support $S_p = \{i : p_i \neq 0\}$ and q has support $S_q = \{j : q_j \neq 0\}$. How can we compute the Nash equilibrium? Let's determine requirements for the best response of these players.
For player 1:

$$\forall i \in S_p, \forall l \in A$$
$$e_i A q^T \geq e_l A q^T.$$

Where e_i is unit vector for i=1.
If $i, l \in S_p$ then $e_i A q^T = e_l A q^T$.
For player 2:

$$\forall j \in S_q, \forall l \in B$$
$$p B e_j^T \geq p B e_l^T$$

	$\Sigma q_j = 1$	$\Sigma p_i = 1$	
$j \in S_q$	$q_j > 0$	$p_i > 0$	$i \in S_p$
$j \notin S_q$	$q_j = 0$	$p_i = 0$	$i \notin S_p$

To find NE we only need to solve the following system of constraints:

$j \in S_q$	$pBe_j^T = v$	$e_i A q^T = u$	$i \in S_p$
$k \notin S_q$	$q_k = 0$	$p_k = 0$	$k \notin S_p$
	$\Sigma q_j = 1$	$\Sigma p_i = 1$	

There are 2(N+1) equations and 2(N+1) variables.
Unique solution requires non-degenerate system of constraints. Otherwise there is an infinite number of Nash equilibria.
Algorithm: For all possible subsets of supports:

- Check if the corresponding linear programming has feasible solution (using e.g. simplex);
- If so, STOP : the feasible solution is Nash equilibrium.

Question: How many possible subsets supports are there to try?
Answer: At most $2^n \cdot 2^n = 4^n$. The algorithm finds all the Nash equilibria. So, unfortunately, the algorithm requires worst-case exponential time.

The following is an example of the game with an exponential number of Nash equilibria.

$$u_2(i,j) = u_1(i,j) = \begin{cases} 1 & \text{if i=j} \\ 0 & \text{otherwise} \end{cases}$$

For each support, a uniform distribution for both players is Nash Equilibrium.

5.4 Approximate Nash Equilibrium

5.4.1 Algorithms for approximating equilibria

Definition $\wp^*$ is ϵ-Nash if for every player p* and every (mixed) strategy $a_j \in A_i$,

$$u_i(a_j, p^*_i) \leq u_i(a_j, p^*_i) + \epsilon$$

If $\epsilon = 0$ then this is a general Nash equilibrium.

Theorem 5.1 *For any two-player game G, there exists an ϵ-Nash, whose support's size is $\frac{12}{\epsilon^2} logn$.*

Proof: The proof is based on the probabilistic method. For the present we assume that all utilities are between 0 to 1. By Chernoff bound, for every i.i.d.$x_1, x_2, ...x_n$ s.t. $E[x_i] = p$. We have

$$Pr[|\overline{x} - p| \geq \epsilon] \leq e^{-2\epsilon^2 l},$$

where

$$\overline{x} = \frac{1}{n} \sum x_i.$$

Let (p,q) be a Nash equilibrium. By sampling each distribution l times we get the distribution ($\hat{p}$,$\hat{q}$). Q.E.D.

We now prove that ($\hat{p}$,$\hat{q}$) is an ϵ-Nash.

Lemma 5.2

$$| \hat{p}A\hat{q}^T - pAq^T | \leq \epsilon$$

$$| \hat{p}B\hat{q}^T - pBq^T | \leq \epsilon$$

Proof: We prove the inequality for A, a similar proof holds for B. Using the Chernoff bound again we get:

$$Pr[\exists i \mid e_i A\hat{q}^T - e_i Aq^T | \geq \epsilon]$$

$$\leq NPr[| e_i A\hat{q}^T - e_i Aq^T | \geq \epsilon]$$

$$\leq Ne^{-2\epsilon^2 l}$$

Where N is the number of actions.
We define the random variable:

$$Q_j^i = \{(l_i A)_j = a_{ij} w.p \cdot q_j\}$$

$$E[Q_j^i] = e_i A q^T$$

$$\frac{1}{l}\sum Q^i = e_i A \hat{q}^T$$

The bound of the error probability is $Ne^{-2\epsilon^2 l}$. Perform this action for $(B,p)(B,q)(A,p)(A,q)$. Hence, with probability $4Ne^{-2\epsilon^2 l}$ this takes place for $\hat{q}$ and $\hat{p}$. Show that we receive the ϵ-Nash equilibrium.

$$\mid pAq^T - \hat{p}A\hat{q}^T \mid \leq$$

$$\mid pAq^T - pA\hat{q}^T \mid + \mid pA\hat{q}^T - \hat{p}A\hat{q}^T \mid \leq 2\epsilon$$

Since (p,q) is Nash equilibrium, $e_i Aq^T \leq pAq^T$. From the approximation we get $e_i A\hat{q}^T \leq pAq^T + \epsilon \leq \hat{p}A\hat{q} + 3\epsilon$, with success probability of $2Ne^{-2\epsilon^2 l}$. We choose $\epsilon' = \frac{\epsilon}{3}$ for $l \geq \frac{9}{2}\frac{1}{\epsilon'^2}\ln 2N$. The success probability is positive. This would mean that there are exist $\hat{q}$ and $\hat{q}$ satisfy all conditions. Hence, $(\hat{q},\hat{q})$ is ϵ-Nash equilibrium. $\Omega(\frac{1}{\epsilon^2}\ln N)$ Q.E.D.

Algorithm: For all groups whose size is at most $\frac{1}{\epsilon^2}\ln N$

- compute ϵ-Nash equlibrium for all pairs of groups.

5.5 Hardness results for Nash Equilibrium

In this section we prove that many problems related to Nash Equilibrium are NP-hard to compute.

5.5.1 Proof Outline

We construct a 2 Player game G. Our decidability problem is the following: Is there an expected Nash equilibrium in G, where both players have an expected utility of 1. We use a reduction from the well known NP-hard problem SAT to this decidability problem. Afterwards we elaborate on this result to illustrate that many Nash Equilibrium problems about 2 player games are NP-hard to compute.

5.5.2 Definitions

- Let θ be a boolean CNF (Conjunctive Normal Form) formula.
- Let V be the set of variables $v_i \in \theta$. $|V| = n$.
- Let L be the set of all possible literals l_i composed from the variables in V. Thus for every $x_i \in V$ there exists $x_i, \bar{x}_i \in L$ and $|L| = 2n$.
- Let C be the set of clauses whose conjunction is θ.
- Let $getVar(\cdot)$ be a function that returns for any literal $l \in L$ the variable appearing in the literal, i.e. $getVar(x_i) = getVar(\bar{x}_i) = x_i$.

- Let f be an arbitrary symbol.
- Let $G(\theta)$ be a 2-Player game. In this game each player selects one of the following: a variable, a literal, a clause from θ or the symbol f. This constitutes his sole action in the game. Game G's outcomes are defined in table 5.1

Thus $\Sigma_1 = \Sigma_2 = V \cup L \cup C \cup \{f\}$

	$\ell_2 \in L$	$v_2 \in V$	$c_2 \in C$	f
$\ell_1 \in L$	$(1,1)$ $\ell_1 \neq \ell_2$ $(-2,-2)$ $\ell_1 = \bar{\ell_2}$	$(-2,-n+2)$ $v(\ell_1) = v_2$ $(-2,+2)$ $v(\ell_1) \neq v_2$	$(-2,+2)$ $\ell_1 \notin c_2$ $(-2,-n+2)$ $\ell_1 \in c_2$	$(-2,1)$ $(-2,1)$
$v_1 \in V$	$(+2,-2)$ $v(\ell_2) \neq v_1$ $(-n+2,-2)$ $v(\ell_2) = v_1$	$(-2,-2)$	$(-2,-2)$	$(-2,1)$
$c_1 \in C$	$(+2,-2)$ $\ell_2 \notin c_1$ $(-n+2,-2)$ $\ell_2 \in c_1$	$(-2,-2)$	$(-2,-2)$	$(-2,1)$
f	$(1,-2)$	$(1,-2)$	$(1,-2)$	$(0,0)$

Table 5.1: Definiton of Game $G(\theta)$

5.5.3 Reduction Proof

Lemma 5.3 *If θ is satisfiable then there exists a (1,1) Nash equilibrium in G*

Proof: If θ is satisfiable then there are $l_1, l_2, ..., l_n \in L$ (where $getVar(l_i) = x_i$) that when assigned the value $True$ (by setting the value of the underlying variable to the satisfying assignment) satisfy θ. If the other player plays all of these l_i with probability $\frac{1}{n}$ then playing the same strategy as well with the same probability yields a utility of 1. We argue this is a Nash Equilibrium.
We show that neither players can net a higher utility by changing his strategy and thus, by definition, it is a Nash Equilibrium:

- Playing the negation of one of the l_i's gives an expected utility of:

$$\frac{1}{n}(-2) + \frac{n-1}{n}(1) < 1$$

- Playing a $v_i \in V$ yields a utility of:

$$\frac{1}{n}(2-n) + \frac{n-1}{n}(2) = 1$$

  The reason is that there is a probability of $\frac{1}{n}$ that the other player chose an l_i such that $getVar(l_i) = x_i$.

- playing a clause $c \in C$ gives a utility of:

$$\frac{1}{n}(2-n) + \frac{n-1}{n}(2) = 1$$

  This happens since each l_i is a member of one or more of the clauses.

- Finally, by choosing f the utility is also 1.

Q.E.D.

This is the only Nash Equilibrium in the game that has an expected utility of 1 to both players. Actually the game has only one more Nash Equilibrium where both players put probability 1 on f and get $(0, 0)$.

Proof:

1. It is easy to verify there are no equilibriums where one player plays purely f and the other does not.

2. Assume both play a mixed strategy where the probability of playing f is 0. The maximal joint utility $(u_1 + u_2)$ in the game is 2. If any player has expected utility less than 1 than he is better off switching to playing f with probability 1. Thus (1,1) is also the maximal joint utility in any equilibrium. If either player plays V and C with a positive probability it follows that the joint utility is below 2. By the linearity of expectation it follows that at least one player has utility below 1. This player is better off playing purely f and thus V or C are never played in a Nash Equilibrium.

3. Thus we can assume both players put positive probabilities on strategies in $L \cup \{f\}$. If one player puts positive probability on f then the other player is strictly better off playing purely f since, like playing L, it yields 1 when the other player plays L and it performs better than L when the other player plays f. It follows that the only equilibrium where f is ever played is the one when both players play purely f.

4. Now we can assume both player only put positive probabilities on elements of L. Suppose that for some $l \in L$, the probability that player 1 plays either l or $\bar{l}$ is less than $\frac{1}{n}$ then the expected utility of the player 2, playing $v \in V$ such that $v = getVar(\bar{l})$, is:
$$u_2 > \frac{1}{n}(2 - n) + \frac{n - 1}{n}(2) = 1$$
Hence, this cannot be a Nash Equilibrium. Thus we can assume that for any $l \in L$ the probability that a given player plays l or $\bar{l}$ is precisely $\frac{1}{n}$.

5. If there is an element of L such that player 1 plays with a positive probability and player 2 plays with a positive probability its negation, then both players have expected utilities of less than 1 and thus are better off playing purely f. Thus, in a Nash Equilibrium, if player 1 plays l with some probability, player 2 must play l with probability $\frac{1}{n}$ and thus player 1 must also play l with probability $\frac{1}{n}$. Thus we can assume that for any variable exactly one of its literals is played by both players with a probability of $\frac{1}{n}$. It follows that in any Nash Equilibrium (besides the one where both players play purely f), literals that are played indeed correspond to an assignment to the variables.

Q.E.D.

Lemma 5.4 *If θ is not satisfiable then a Nash Equilibrium with expected utilities of (1,1) doesn't exist in G*

Proof: As we verified above, when θ is satisifiable, the only Nash Equilibrium where the utility is 1 for both players is when both players choose l_i with probability $\frac{1}{n}$. But when θ is not satisfiable, this is not a Nash Equilibrium anymore:

- Let $c \in C$ be a clause that is not satisfied by the assignment, that is, none of its literals are ever played. Thus playing c nets a utility of 2 and each player is better off switching to this strategy.

Q.E.D.

5.5.4 Conclusions and Corollaries

Thus it is NP-hard to determine whether a certain equilibrium exists in a game. Also, since $(1, 1)$ is the equilibrium that maximizes the "social welfare" (combined utility of both players) it can be viewed as the optimal Nash Equilibrium of the game. Thus finding the optimal Nash Equilibrium of the game is also NP-hard. Using the same game G, depending on how we state the decidability problem, we can deduce that many other Nash Equilibrium related problems are NP-hard. For example the number of different Nash Equilibrium in the game, is there an Nash Equilibrium where all players are guaranteed at least k and so on. Thus information regarding the Nash Equilibrium of games is generally hard to compute.

5.6 Brouwer's fixed point theorem

Theorem 5.5 *(Brouwer) Let $f : S \to S$ be a continuous function from a non-empty, compact, convex set $S \in \Re^n$ into itself, then there is $x^{\cdot} \in S$ such that $x^{\cdot} = f(x^*)$ (i.e. $x^{\cdot}$ is a fixed point of function f).*

5.6.1 Proof Outline

For $\Re^1$ the proof is a simple one is proved directly. For the two dimensional case we prove the theorem on triangles, aided by Sperner's lemma (which we will prove as well). Since we can "cut out" a triangle out of any convex, compact set, the theorem holds for any such set in $\Re^2$. Generalization of the theorem for a triangle in $\Re^n$ follows, but will not be shown here.

5.6.2 In One Dimension

Let $f : [0, 1] \longrightarrow [0, 1]$ be a continuous function. Then, there exists a fixed point, i.e. there is a x^* in $[0, 1]$ such that $f(x^*) = x^*$. There are 2 possibilities:

1. If $f(0) = 0$ or $f(1) = 1$ then we are done.

2. If $f(0) \neq 0$ and $f(1) \neq 1$. Then define:

$$F(x) = f(x) - x$$

In this case:

$$F(0) = f(0) - 0 = f(0) > 0$$
$$F(1) = f(1) - 1 < 0$$

Thus $F : [0,1] \longrightarrow \Re$ where $F(0) \cdot F(1) < 0$. Since $f(\cdot)$ is continuous, $F(\cdot)$ is continuous as well. By the *Intermidate Value Theorem*, there exists $x^* \in [0,1]$ such that $F(x^*) = 0$. By definition of $F(\cdot)$:

$$F(x^*) = f(x^*) - x^*$$

And thus:

$$f(x^*) = x^*$$

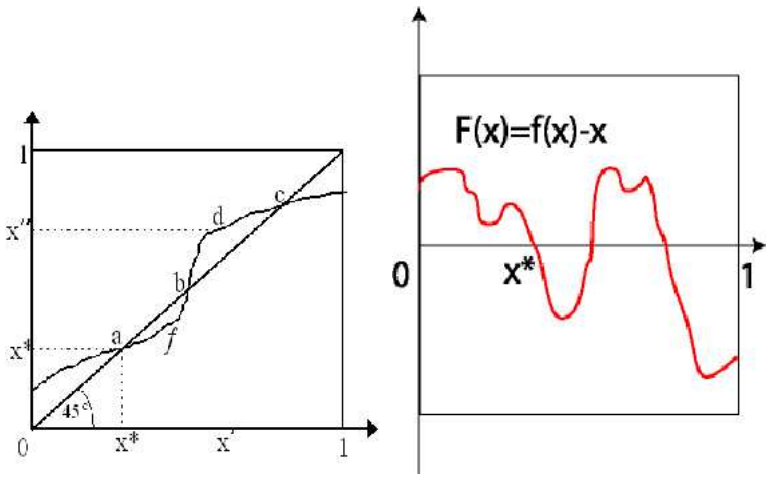


Figure 5.2: A one dimensional fixed point (left) and the function $F(\cdot)$ (right)

5.6.3 In Two Dimensions

Sperner's Lemma

Lemma 5.6 *(Sperner's Lemma) Given a triangle and a triangulation of it into an arbitrary number of "baby triangles".*

1. *Mark the vertices of the original triangle by 0,1,2.*
2. *Mark all other vertices with one of the labels according to the following rule:*
   - *If a vertex lies on the edge of the original triangle, label it by one of the numbers at the end points of that edge.*

- *If a vertex is inside the original triangle label it any way you like.*

There exists a baby triangle that has all three of the labels (one vertex is 0, the second is 1 and the third is 2). In fact there is an odd number of such triangles.

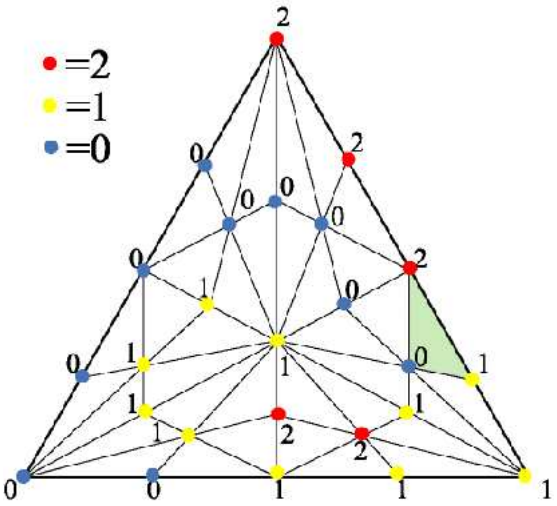


Figure 5.3: Sperner labeled triangle

Proof of Sperner's lemma in one dimension

Given a line segment whose endpoints are labeled 0 and 1 is divided to subsegments, each interior point is labeled is 0 or 1.

Definition A segment is called *completely labeled* if it has a label of 0 at one end and a label of 1 at the other end

- C = number of completely labeled subsegments.
- Z = number of subsegments labeled with 0 and 0.
- O = number of occurrences of 0 at an endpoint of a subsegment - *an odd number* (since for every 0 you label two edges are added and you start off with one such edge)
- $O = 2Z + C$

Thus, C must be odd numbered (and thus $\neq$ 0s)

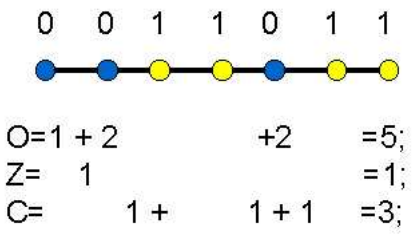


Figure 5.4: Sperner labeled segment

Proof of Sperner's lemma in two dimensions

Definition A triangle is called *completely labeled* if it has a label of 0 at one vertex, a label of 1 at another vertex and a label of 2 at the third vertex

- C = number of baby triangles that are completely labeled
- Z = number of baby triangles with 0 and 1 but not 2 (with 0,1,0 or with 0,1,1)
- O = number of occurrences of 0,1 at an edge of a baby triangle *an odd number*

  1. All occurrences of 0,1 for an interior baby triangle are paired up (since each interior edge is shared by two baby triangles ...)- an even number.
  2. All occurrences of 0,1 for a side baby triangle occur along the base of the original triangle, and thus are an odd number by the one dimension argument.

- $O = 2Z + C$
  Since each triangle in Z has two edges labeled 0,1 it contributes two edges to O. In sum, $2Z$ edges are contributed by all triangles in Z. This suggests the remaining edges in O come from triangles in C. Each triangle in C has one edge labeled 0,1 and thus contributes one edge to O.

Since O is odd and $2Z$ is even, C must be odd numbered (and thus $\neq 0$); which implies there exists at least one completely labeled baby triangle.

Proof of Brouwer's theorem

We can find a triangle in any convex, compact two-dimensional shape. Define the three vertices of the original triangle as A,B and C. Each point p in the triangle will be represented by its barycentric coordinates:

$$p = x_0A + x_1B + x_2C; x_0 + x_1 + x_2 = 1$$

Thus p can be represented as $p = (x_0, x_1, x_2)$.

Label all points of the triangle as follows:

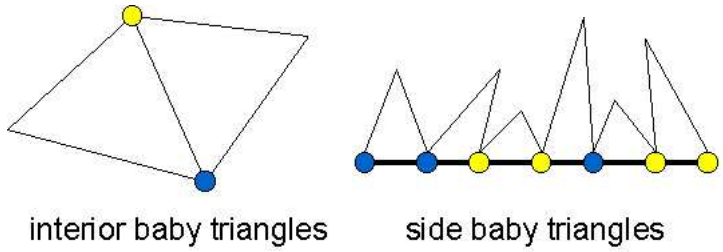


Figure 5.5: Labeling baby triangles in two dimensions

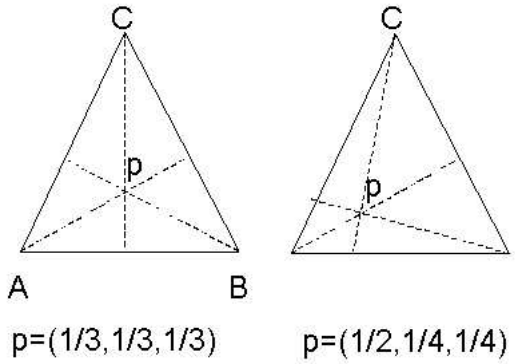
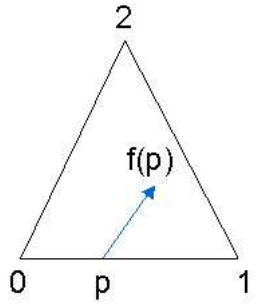


Figure 5.6: Sample barycentric coordinates of point p

- If $p = (x_0, x_1, x_2)$ and $f(p) = (x_0^{'}, x_1^{'}, x_2^{'})$ inspect the coordinates of p and $f(p)$ until you find the first index $i \in \{0, 1, 2\}$ such that $x_i^{'} < x_i$, label p by the label i.
- If it happens that for some point p there is no strict inequality like that, it must be that $x_0^{'} = x_0$, $x_1^{'} = x_1$, $x_2^{'} = x_2$. Thus p is a fixed point and we are done.
- According to this rule, the vertices A,B,C are labeled by 0,1,2.
- Each point of the edge 0,1 is marked by either 0 or 1. Similar statements hold for the other edges.
- Divide the triangles into smaller and smaller triangles with diameters approaching 0. At each step label the triangles by the rule above. Labeling is as in Sperner's Lemma.
- For each subdivision there exists at least one triangle labeled with all three labels.
- As we divide to smaller and smaller triangles, the vertices of the baby triangles that are completely labeled must eventually converge to some point $q = (y_0, y_1, y_2)$ (see figure 5.8). Now we use the labeling scheme we developed earlier and the completely labeled property of q that assures us that, due



the coordinates of p are $(*, *, 0)$ and they transformed by f into $(*, *, *)$; the x_2 increases, thus the label of p cannot be 2, thus it must be either 0 or 1;

Figure 5.7: Example of how a labeling of a point p is determined

to continuity, for q and $f(q)$ we must have:

$$y_0^{'} \leq y_0;\ y_1^{'} \leq y_1;\ y_2^{'} \leq y_2$$

Since the barycentric coordinates of a point add up to 1, these must be equalities:

$$y_0^{'} = y_0;\ y_1^{'} = y_1;\ y_2^{'} = y_2$$

And thus $f(q) = q$ and q is a fixed point. The proof for n-dimensional spaces is similar using an n-dimensional triangle.

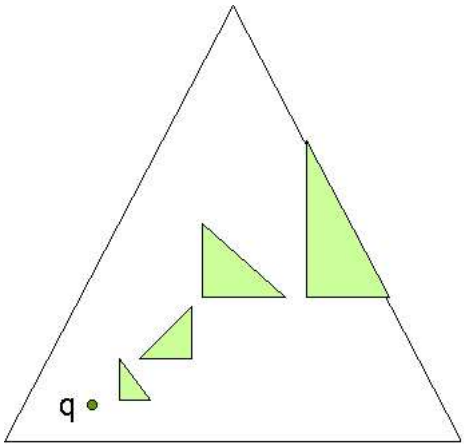


Figure 5.8: Dividing into smaller and smaller triangles - approaching to point q

Lecture 6:Congestion and potential games

Lecturer:Yishay Mansour *Scribe:Nir Yosef, Ami Koren*

6.1 Lecture overview

So far we've seen that not every strategic game has a deterministic *Nash* equilibrium.
In this lecture we discuss a certain class of games: congestion and potential games, for which we prove the existence of a deterministic *Nash* equilibrium.
In the coming sections we define the above classes, show the relation between them and estimate the complexity of finding a deterministic *Nash* equilibrium for a potential game.

6.2 Congestion game

6.2.1 Example

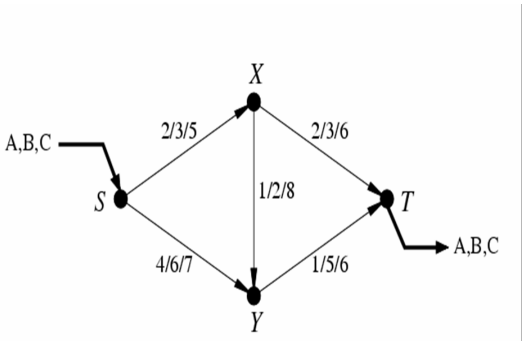


Fig.1 Example of a congestion game

Let us start with an illustrative example: In the model described above, Players A,B and C have to go from point S to T using road segments SX,XY,...etc. Numbers on edges denote the cost for a single user for using the corresponding road segment, where the actual cost is a function of the actual number of players using that road segment(i.e. a *descrete delay* function). For example: if segment SX is used by a 1,2, or 3 users, the cost on that segment would be 2,3, or 5, respectively. The total cost of each player is the sum of all segments he uses. Note that the players are therefore engaged in a game which can be represented in a strategic form(as a cost matrix).

6.2.2 Congestion game - Definition

A congestion model (N,M,$(A_i)_{i\in N}$,$(c_j)_{j\in M}$) is defined as follows:

- $N = \{1..n\}$ denotes the set of players
- $M = \{1..m\}$ denotes the set of facilities
- For $i \in N$, A_i denotes the set of strategies of player i, where each $a_i \in A_i$ is a non empty subset of the facilities.
- For $j \in M$, $c_j \in R^n$ denotes the vector of costs, where $c_j(k)$ is the cost related to each user of facility j, if there are exactly k players using that facility.

The *congestion game* associated with a congestion model is a game in strategic form with the set of N players, with sets of strategies $(A_i)_{i\in N}$ and with cost function defined as follows: Let $A = \times_{i\in N} A_i$ be the set of all possible deterministic profiles(players strategy vectors). For any $\vec{a} \in A$ and for any $j \in M$, let $n_j(\vec{a})$ be the number of players using facility j, assuming $\vec{a}$ to be the current profile.
Now, define the overall cost function for player i: $u_i(\vec{a}) = \sum_{j\in a_i} c_j(n_j(\vec{a}))$

Remark 6.1 *All players are equal in a sense that they have the same 'weight'(it doesn't matter* which *players are using a facility, only* how many *players are using it).*

6.2.3 Deterministic equilibrium

Theorem 6.2 *Every finite congestion game has a pure strategy(deterministic) equilibrium.*

Proof: Let $\vec{a} \in A$ be a deterministic strategy vector as defined above,
let Φ: $A \to R$ be a potential function defined as follows: $\Phi(\vec{a}) = \sum_{j=1}^{m} \sum_{k=1}^{n_j(\vec{a})} c_j(k)$
Consider the case where a single player changes its strategy from a_i to b_i(where $a_i, b_i \in A_i$).
Let Δu_i be the change in its cost caused by the the change in strategy:
$\Delta u_i = u_i(b_i, \vec{a_{-i}}) - u_i(a_i, \vec{a_{-i}}) = \sum_{j\in b_i - a_i} c_j(n_j(\vec{a}) + 1) - \sum_{j\in a_i - b_i} c_j(n_j(\vec{a}))$
(explanation: change in cost = cost related to the use of new facilities minus cost related to use of those facilities which are not in use anymore due to strategy change)
Let $\Delta\Phi$ be the change in the potential caused by the change in strategy:
$\Delta\Phi = \Phi(b_i, \vec{a_{-i}}) - \Phi(a_i, \vec{a_{-i}}) = \sum_{j\in b_i - a_i} c_j((n_j(\vec{a}) + 1) - \sum_{j\in a_i - b_i} c_j(n_j(\vec{a}))$
(explanation: immediate from potential function's definition)
Thus we can conclude that for a single player's strategy change we get
$\Delta\Phi = \Delta u_i$.

That's an interesting result: We can start from an arbitrary deterministic strategy vector $\vec{a}$, and at each step one player reduces it's cost. That means, that at each step Φ is reduced identically. Since Φ can accept a finite amount of values, it will eventually reach a local minima. At this point, no player can achieve any improvement, and we reach a *NE*. □

6.3 Potential games

6.3.1 Potential functions

Let $G =< N, (A_i), (u_i) >$ be a game in strategic form and let $A = \times_{i \in N} A_i$ be the collection of all deterministic strategy vectors in G.

Definition A function Φ: $A \to R$ is an *exact potential* for game G if
$\forall_{\vec{a} \in A} \forall_{a_i, b_i \in A_i}\ \Phi(b_i, \vec{a_{-i}}) - \Phi(a_i, \vec{a_{-i}}) = u_i(b_i, \vec{a_{-i}}) - u_i(a_i, \vec{a_{-i}})$

Definition A function Φ: $A \to R$ is a *weighted potential* for game G if
$\forall_{\vec{a} \in A} \forall_{a_i, b_i \in A_i}\ \Phi(b_i, \vec{a_{-i}}) - \Phi(a_i, \vec{a_{-i}}) = \omega_i(u_i(b_i, \vec{a_{-i}}) - u_i(a_i, \vec{a_{-i}})) = \omega_i \Delta u_i$
Where $(\omega_i)_{i \in N}$ is a vector of positive numbers (weight vector).

Definition A function Φ: $A \to R$ is an *ordinal potential* for a minimum game G if
$\forall_{\vec{a} \in A} \forall_{a_i, b_i \in A_i}\ (\Phi(b_i, \vec{a_{-i}}) - \Phi(a_i, \vec{a_{-i}}) < 0) \Leftarrow (u_i(b_i, \vec{a_{-i}}) - u_i(a_i, \vec{a_{-i}}) < 0)$ (the opposite takes place for a maximum game).

Remark 6.3 *Considering the above definitions, it can be seen that the first two definitions are private cases of the third.*

6.3.2 Potential games

Definition A game G is called an *ordinal potential game* if it admits an ordinal potential.

Theorem 6.4 *Every finite ordinal potential game has a pure strategy(deterministic) equilibrium.*

Proof: Similarly to the previous proof, starting from an arbitrary deterministic strategy vector,after a finite number of steps of single player improvement, we will reach a local minima which is, as was explained above, a deterministic equilibrium. □

6.3.3 Examples

Exact potential game

Consider an undirected graph $G = (V, E)$ with a weight function $\vec{\omega}$ on its edges. The goal is to partition the vertices set V into two distinct subsets D_1, D_2(where $D_1 \cup D_2 = V$):
for every player i, choose $s_i \in \{-1, 1\}$ where choosing $s_i = 1$ means that $i \in D_1$ and the opposite for D_2. The weight on each edge denotes how much the corresponding vertices 'want' to be on the same set. Thus, define the value function of player i as $u_i(\vec{s}) = \sum_{j \neq i} \omega_{i,j} s_i s_j$

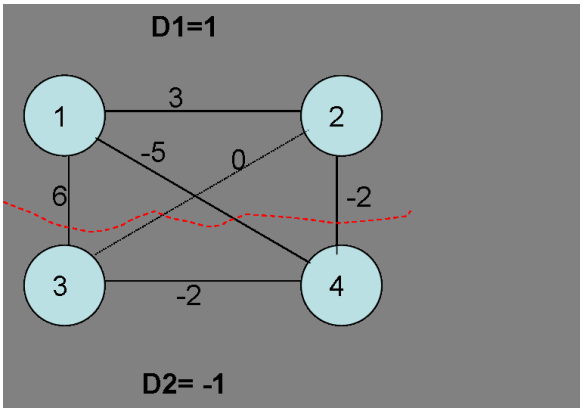


Fig.2 Example for an exact potential game

On the example given in figure 2 it can be seen that players 1,2,4 have no interest in changing their strategies, However, player 3 is not satisfied, it can increase his profit by changing his chosen set to D_1.
Using $\Phi(\vec{s}) = \sum_{j<i} \omega_{i,j} s_i s_j$ as our potential function, let us consider the case where a single player i changes its strategy(shifts from one set to another):
$\Delta u_i = \sum_{j \neq i} \omega_{i,j} s_i s_j - \sum_{j \neq i} \omega_{i,j}(-s_i) s_j = 2 \sum_{j \neq i} \omega_{i,j} s_i s_j =$
$2 \sum_{j:\ j<i} \omega_{i,j} s_i s_j + 2 \sum_{j:\ i<j} \omega_{i,j} s_i s_j = \Delta(\Phi)$
Which means that Φ is an exact potential function, therefore we conclude that the above game is an exact potential game.

Weighted potential game

Consider the following load balancing congestion model(N,M,$(\omega_i)_{i \in N}$) with M identical machines, N jobs and $(\omega_i)_{i \in N}$ weight vector($\omega_i \in R^+$). The load on a machine is defined as the sum of weights of the jobs which use it: $L_j(\vec{a}) = \sum_{i:\ a_i = j} \omega_i$ where $\vec{a} \in [1..M]^N$ is a deterministic strategy vector.
Let $u_i(\vec{a}) = L_{a_i}(\vec{a})$ denote the cost function of player i. We would like to define a potential function whose reaction to a single player's strategy change will be correlated with the reaction on the player's cost function.
The potential function is defined as follows: $\Phi(\vec{a}) = \sum_{j=1}^{M} \frac{1}{2} L_j^2$, consider the case where a single job shifts from its selected machine M_1 to another machine M_2 (where M_1 and M_2 are two arbitrary machines):
Let Δu_i be the change in its cost caused by the strategy change:
$\Delta u_i = u_i(M_2, \vec{a_{-i}}) - u_i(M_1, \vec{a_{-i}}) = L_2(\vec{a}) + \omega_i - L_1(\vec{a})$
(explanation: change in job's load = load on new machine minus load on old machine)
Let $\Delta\Phi$ be the change in the potential caused by the strategy change:
$\Delta\Phi = \Phi(M_2, \vec{a_{-i}}) - \Phi(M_1, \vec{a_{-i}}) = \frac{1}{2}[(L_1(\vec{a}) - \omega_i)^2 + (L_2(\vec{a}) + \omega_i)^2 - L_1^2(\vec{a}) - L_2^2(\vec{a})] =$
$= \omega_i(L_2(\vec{a}) - L_1(\vec{a})) + \omega_i^2 = \omega_i(L_2(\vec{a}) + \omega_i - L_1(\vec{a})) = \omega_i \Delta u_i$

Therefore, we can conclude that the model at hand is a weighted potential game.

General(ordinal) potential game

Consider the following load balancing congestion model$(N,M,(\omega_{i,j})_{i\in N,j\in M})$ with M related machines, N jobs and $\vec{\omega}$ a machine dependent weight vector(where $\omega_{i,j} \in N^+$ is the weight of job i having been assigned to machine j).
Here we have similar definitions to those we have presented in the above example:
Load on a machine j: $L_j(\vec{a}) = \sum_{i:\ a_i=j} \omega_{i,j}$
Cost function for player i: $u_i(\vec{a}) = L_{a_i}(\vec{a})$
The potential function will now be defined as $\Phi(\vec{a}) = \sum_{j=1}^{M} 4^{L_j(\vec{a})}$
Consider the case where a single job shifts from its selected machine M_1 to another machine M_2. Change in players' cost is calculated in a similar fashion to the above example:
$\Delta u_i = L_2(\vec{a}) + \omega_{i,2} - L_1(\vec{a})$
Change in the potential caused by the strategy change will now be:
$\Delta\Phi = 4^{L_1(\vec{a})-\omega_{i,1}} + 4^{L_2(\vec{a})+\omega_{i,2}} - 4^{L_1(\vec{a})} - 4^{L_2(\vec{a})}$
If $\Delta u_i < 0 \Rightarrow L_2(\vec{a}) + \omega_{i,2} < L_1(\vec{a}) \Rightarrow L_2(\vec{a}) + \omega_{i,2} + 1 \leq L_1(\vec{a})$.
In addition $L_1(\vec{a}) - \omega_{i,1} + 1 \leq L_1(\vec{a})$ (both conclusions under the assumption that$(\omega_{i,j} \in N^+)$). From the two inequalities we conclude:
$\{4^{L_1(\vec{a})-\omega_{i,1}} \leq 4^{L_1(\vec{a})-1}, 4^{L_2(\vec{a})+\omega_{i,2}} \leq 4^{L_1(\vec{a})-1}\} \Rightarrow \Delta\Phi \leq 2\cdot 4^{L_1(\vec{a})-1} - 4^{L_1(\vec{a})} = -2\cdot 4^{L_1(\vec{a})-1} < 0.$
Therefore, we can conclude that the model at hand is a general potential game.

Another example of a general potential game is given by taking the same model we have described as an example for an exact potential game, along with a slightly different value function $u_i(\vec{s}) = SIGN(\sum_{j\neq i} \omega_{i,j} s_i s_j)$ and the same potential function.
Following a similar proof it can be seen that this time we get an ordinal potential.

6.3.4 Finite improvement path

We use the concept of a Finite improvement path in order to define an abstract ordinal potential function for a given strategic game. A finite improvement path is defined as follows:
For $G =< N, (A_i), (u_i) >$ minimum game in strategic form, and $A = \times_{i\in N} A_i$ collection of all deterministic strategy vectors let $\Pi = (V, E)$ be a graph such that
$V = A$ and $E = \{< \vec{a}, \vec{b} >\in A^2$: $\exists i\ [(b_i, \vec{a_{-i}}) = \vec{b}\ ] \wedge [u_i(\vec{b}) < u_i(\vec{a})]\}$

Lemma 6.5 *If* Π *is acyclic then the corresponding game* G *possesses a deterministic equilibrium.*

Proof: Every acyclic graph has a sink(vertex without outgoing edges). Every sink on Π is a deterministic equilibrium(follows immediately from the definition of E). □

Now let us define the potential function $\Phi(\vec{a})$ as the length of the longest route on Π starting from $\vec{a}$. Note that going from one vertex to another on Π(which is equivalent to a step where a single user changes its strategy, thus reducing its cost) will reduce the value of Φ (immediate from the definition of Φ). In addition - the number of such steps is final(because G is final).
Having said that and from the definition at **8.3.1** we can conclude that Φ is an ordinal potential function.
Every path on that graph is an improvement path with respect to the above potential function. The last vertex of every such path corresponds to an equilibrium point, as was explained in the proof above.

6.4 Computing equilibrium in congestion games

We have seen on proof of theorem **8.2** that every general congestion game has an exact potential function. We can use that potential function in order to find an equilibrium point(by following the same scheme as described on the proof of theorem **8.2**). The problem is that the number of steps might be exponential in the size of the game.

6.4.1 Symmetric network's game

(An example for computing equilibrium using reduction)
A symmetric network's game NG is defined as follows: given a graph $G = (V, E)$ with source and destination vertices (S, T), the players have to choose a route on G leading from S to T. Each edge has a delay value which is a function of number of players using it.
Now, let us look at the full definition of NG as a congestion game $(\mathrm{N},\mathrm{E},(A_i)_{i\in N},(c_e)_{e\in E})$:

- N denotes the set of players
- E denotes the set of edges of G
- A_i is the set of strategies of player i, where each $a_i \in A_i$ is a route on G leading from S to T
- For $e \in E$, $c_e \in R^n$ denotes the vector of delays, where $c_e(k)$ is the delay related to edge e, if there are exactly k players using that edge.
- Player's cost function is $u_i(\vec{a}) = \sum_{e\in a_i} c_e(n_e(\vec{a}))$ (where n_e as before denotes the number of players using edge e)

Remark 6.6 *On proof of theorem 8.2 we saw that for this kind of game the potential function* $\Phi(\vec{a}) = \sum_{e=1}^{m}\sum_{k=1}^{n_e(\vec{a})} c_e(k)$ *is exact.*

reduction to min-cost flow

considering the graph $G = (V, E)$ and the delay functions $\{c_e\}_{e \in E}$, we replace in G each edge e with n parallel edges between the same nodes, each with capacity 1, and with costs $c_e(1), .., c_e(n)$.

Lemma 6.7 *The cost of a full min-cost flow of* n *units on the new network is equal to* $\Phi(\vec{a})$ *where* $\vec{a}$ *is a strategy vector corresponding to that flow.*

Proof: Let f be a full flow of n units.f can be divided into n distinct routes(because the capacity of each edge is exactly 1). Consider every route as the strategy of a single player of the game *NG*. We define the corresponding strategy vector $\vec{a}$ as some ordered collection of all these routes(it is not important in which order we define $\vec{a}$ because all players are equal). Since f is minimal then it will first use the cheaper edges, therefore the contribution of a collection of edges $e_1, .., e_n$ on the new network which corresponds to single edge e on G will be $\sum_{k=1}^{n_e(\vec{a})} c_e(k)$ and the total cost of f is thus $\sum_e \sum_{k=1}^{n_e(\vec{a})} c_e(k) = \Phi(\vec{a})$

□

Remark 6.8

- *Any minima on* Φ *is an equilibrium point(Immediate from the definition of* Φ*).*
- *It is easy to see that for any integer min-cost flow in the new network, the strategy vector corresponding to that flow minimizes* Φ*.*

Corollary 6.9 *For every min-cost flow on the network defined above, the corresponding strategy vector is an equilibrium point.*

Corollary 6.10 *There is a polynomial algorithm for finding a pure equilibrium in symmetric network congestion games.*

6.4.2 PLS class

We saw that we can always find deterministic equilibrium in general congestion game. We also saw that in some cases, we have polynomial algorithm for doing that. How hard it is to find the equilibrium? We will now show, that in some cases, the problem becomes exponential. In order to do this, we will define a class for local-optima search.

Definition PLS class(Polynomial-time Local Search)
Terminology:

- I - Collection of possible inputs(graphs, for example)

- F_x - For each instance $x \in I$ we have a finite set F_x of the possible solutions, all with the same polynomially bounded length(For example, all *TSP* paths at each graph).
- $c(x, s)$ - Cost function of a solution $s \in F_x$ given an instance $x \in I$. For example- for each graph $x \in I$ and path $s \in F_x$, $c(x, s)$ marks the cost of the path(For $s \notin F_x$ $c()$ returns "illegal")
- $N_x(s) \subseteq F_x$ - Neighborhood function. This function defines the environment of each possible solution.
- both $N_x(s)$ and F_x are recognizable in polynomial time

The problem: finding a local optimum solution. That is, to find $x \in I$ and $s \in F_x$, such that $\forall \grave{s} \in N_x(s) : \ c(x, \grave{s}) \geq c(x, s)$

For complete definition of PLS, see [3].

Sample for generic problem

- $I = \{0, 1\}^n$
- $F_x = I$ -(has no meaning for solution)
- $N_x = \{y | H(x, y) = 1\}$ - The set of neighbors of each vector, defined as the set of all vectors which differ from it in exactly one bit(Hamming distance of 1)
- $c(x)$ - Some generic cost function

This problem can be thought of as seeking for a local minimal-cost vector among the set $\{0, 1\}^n$ where locality is defined by hamming distance.
It can be shown that there are cost functions for which there exists an improvement path at length $\frac{1}{4}2^n \leq l \leq 2^n$(*Snake in a box*). This means that the improvement path has to go through a significant number of the edges of the box in order to reach a local optima.

The complexity of the problem is cost-function depended. For example, if we define cost function c(x) : $I \rightarrow \{0, 1\}$, we have an easy 1-step algorithm: Check all your neighbors, if they're all 1 - exit, else - go to the 0-neighbor and exit. The problem is getting hard when there are a lot of different costs, since the improvement at each step can be much smaller then the range of the costs.

PLS-Complete problems
Definition $L \in PLS$ is **PLS-complete** when every $\Pi \in PLS$ is reducible to L

Here are some of the problem which are known to be PLS-Complete:

- Local minima in a cube(See above)
- MAX-CUT - Find the edges that give maximal weight cut of graph vertices - In the local version, we consider two cuts as neighbors, if they differ in exactly one vertex
- W2SAT - Try to satisfy as much clauses as possible(in cases that the expression is not satisfiable). In the local-version, we consider two solutions as neighbors if they differ in the value of exactly one variable(i.e. our goal is to reach a solution where a single switch of variable will not improve the cost)
- TSP_2 - Local version of Traveling Sales Person, where 2 routes are considered as neighbors if they differ in at most 2 edges

Theorem 6.11 *A general congestion game, symmetric congestion game, and asymmetric network game are all PLS-complete*

Proof:
We will show that a generic congestion game is PLS-complete. We will do it by a reduction from a known PLS-complete problem:

Weighted not-all-equal 3SAT is a PLS-complete problem described as follows:

- **Input** - Monotonic 3-CNF. A clause is considered satisfied if it is not all-1 and not all-0. There is a weight for each clause
- **Output** - An assignment that satisfies maximal weight of clauses
- **Cost** - Sum of weights on all unsatisfied clauses
- **Locality** - In the local version, we consider two assignments as neighbors if they have different values for exactly one variable

The Reduction
Given an instance of the 3SAT problem above, we build a corresponding congestion game. We want to show that the *Nash* equilibrium at the congestion game is equivalent to local-minima at the above 3SAT problem. Therefore, had we known to find a *Nash* equilibrium at the congestion game, we would have been able to solve our 3SAT problem.

For any given *3-CNF*, we build the corresponding congestion game as follows:

- variable $\longrightarrow$ player

- clause $T_j \longrightarrow$ 2 resources: m_j^0, m_j^1
- action $\longrightarrow$ For a player we have 2 actions:
  - $I = \{m_j^1 |\ x_i \in T_j\}$
  - $II = \{m_j^0 |\ x_i \in T_j\}$

  Explanation: Each player can choose whether to play all on 0, or all on 1
- 3SAT cost function: For $a \in \{0,1\}$, $C_{m_j^a}(1) = C_{m_j^a}(2) = 0, C_{m_j^a}(3) = w_i$ - This cost function punishes clauses with all-equal values. $C_{m_j^a}(0) = 0$, because we already punish it at $C_{m_j^{1-a}}(3)$.
- assignment of $x_i = 1$ $(3SAT) \Longleftrightarrow$ Player's action A_i=I(congestion game) , which means that for every vector of deterministic strategy on the congestion game there exists a corresponding assignment on(3SAT) and vice versa.

After defining the game, we show the equivalence between *Nash* equilibrium at the congestion game to local minima at the our 3SAT problem.

- Change of player's action at game: Assuming a player changed it's action from I to II:
  - D_1 - The clauses that became satisfied as result of the change
  - D_2 - The clauses that became unsatisfied as result of the change

  The gain from the change:

$$\triangle u^i = \sum_{j \in D_2} w_j - \sum_{j \in D_1} w_j$$

- Switching a variable's value at 3SAT will have the same affect:

$$\triangle u^i = \sum_{j \in D_2} w_j - \sum_{j \in D_1} w_j$$

Therefore, *Nash* equilibrium at the congestion game is equivalent to a local-minima at the 3SAT problem

□

6.4.3 ε-*Nash* of congestion game

Finding deterministic *Nash* equilibrium for a general congestion game is hard, but it might be easier in some cases. For instance, for small number of players, or for finding ε-*Nash*.

We now present an algorithm for finding an ε-*Nash* of a congestion game with the following potential function:

$$\phi(\overrightarrow{a}) = \sum_{j=1}^{m} \sum_{k=1}^{n_j} c_j(n_k)$$

We start from an arbitrary deterministic strategy vector $\vec{a_0}$. At each step we decrease ϕ with at least ε. If we can't, we reached the ε-*Nash*. Since for each $\vec{a}$ we have

$$\phi(\overrightarrow{a}) \leq \sum_{j=1}^{m} n_j c_j(n_j) \leq n \cdot m \cdot c_{max}$$

Then the number of steps is at most $\frac{\phi(\overrightarrow{a_0})}{\varepsilon}$, which is limited by $\frac{n \cdot m \cdot c_{max}}{\varepsilon}$

6.5 Equivalence of potential and congestion game

At the beginning of this lecture we saw that each congestion game admits a potential function and therefore it is a potential game. We now show the other direction: For each exact potential game there exists a matching congestion game.

Definition Equivalence of games

Assuming there are 2 games: $< N, (A_1^i)_{i \in N}, (u_1^i)_{i \in N} >$, $< N, (A_2^i)_{i \in N}, (u_2^i)_{i \in N} >$

If there is a 1-1 mapping between game 1 and game 2: $g^i : A_1^i \longrightarrow A_2^i$ such that:

$$u_1^i(a_1, \ldots, a_n) = u_2^i(g^i(a_1), \ldots, g^i(a_n))$$

We say that the games are equivalent.

Theorem 6.12 *Each game with exact potential function has equivalent congestion game*

Proof:

Given a game $< N, (A^i)_{i \in N}, (u_1^i)_{i \in N} >$ with an exact potential function ϕ. For simplicity let us assume that $\forall i \in N : \ l = |A^i|$.

We'll start building an equivalent congestion game $< N, V, (c_j)_{j \in M} >$.

Players: Stay the same

Resources: $V = \{0,1\}^{l \cdot n}$ - Each vector represent a single resource, which gives us a total of $2^{l \cdot n}$ resources.
We treat each of these vectors as n vectors, each of size l.

Holding resources: Each player i that play action j, holds the resources with the appropriate bit set: $B_j^i = \{v : \ v_{ij} = 1\}$ (therefore, $|B_j^i| = \frac{2^{ln}}{2}$).

We now seek for a cost function on the resources of the new congestion game, such that the cost u_2^i for each player will satisfy: $u_2^i(a_1, \ldots, a_n) = u_1^i(a_1, \ldots, a_n)$ (according to the definition above).
We define the cost function c for specific resources that will sum to the desired cost for each player. All other resources will have a cost=0.

Given strategy vector $\vec{a}$, then for the vector $v^{\vec{a}}_{i',j'} \in \{0,1\}^{l*n}$ such that:

$$v^{\vec{a}}_{i',j'} = \begin{cases} 1 & j' = a_{i'} \\ 0 & otherwise \end{cases}$$

There is a congestion of exactly n(because it is held by all players). For this vector, we define: $c_{v^{\vec{a}}}(k) = \phi(\overrightarrow{a})$ for $k = n$ and 0 otherwise.

If these resources were the only ones with non-zero-cost, then each player would have a cost of $\phi(\vec{a})$. Therefore, we have to fix it for each player i, by $u_1^i(\overrightarrow{a}) - \phi(\overrightarrow{a})$.

In order to do this, we find a resource $r^i \in V$ that only player$i$ holds. For $\vec{a}$, we define:

$$r^i_{i',j'} = \begin{cases} 1 & i' = i \\ 0 & i' \neq i, j' \in a_{i'} \\ 1 & \text{Otherwise} \end{cases}$$

Meaning: r^i is 1 anywhere, except at the locations corresponding to the actions of the rest of the players
The cost we define for these resources are:

$$c_{r^i}(1) = u^i(\overrightarrow{a}) - \phi(\overrightarrow{a})$$

We have to show that $c_{r^i}(1)$ is well defined for player i. That is, that it doesn't change if the player changes it's action. We do it by using the fact that ϕ is an exact potential, and therefore, for actions a_i, b_i of player i:

$$u_1^i(a_i, a^{-i}) - \phi(a_i, a^{-i}) = u_1^i(b_i, a^{-i}) - \phi(b_i, a^{-i})$$

$$\Rightarrow u_1^i(a_i, a^{-i}) - u^i(b_i, a^{-i}) = \phi(a_i, a^{-i}) - \phi(b_i, a^{-i})$$

$$\Rightarrow \Delta u_1^i(a_i \to b_i) = \Delta\phi(a_i \to b_i)$$
$$\Rightarrow \Delta c_{r^i} = \Delta u_1^i(a_i \to b_i) - \Delta\phi(a_i \to b_i) = 0$$

All other resources have cost=0.

The cost of player i:
$u_2^i(\vec{a}) = \sum_{j \in a_i} c_j(n_j(\vec{a})) = c_{v^{\vec{a}}}(n) + c_{r^i}(1) = \phi(\vec{a}) + u^i(\vec{a}) - \phi(\vec{a}) = u^i(\vec{a})$

Therefore, by the definition above, the potential game is equivalent to the congestion game we've built.

Notice, that the number of resources is exponential in this representation. That means that a general potential game might be much harder then a congestion game.

□

6.6 Bibliography

[1] *The Complexity of Pure Nash Equilibria*, A. Fabrikant, C. Papadimitriou, K. Talawar
[2] *Potential Games*, D. Monderer, L. Shapley, 1994
[3] *How Easy Is Local Search?*, D. Johnson, C. Papadimitriou, M. Yannakakis, 2004, http://faculty.cs.tan

Computational Game Theory **Spring Semester, 2003/4**

Lecture 7: May 4

Lecturer: Yishay Mansour *Scribe: Gur Yaari, Idan Szpektor*

7.1 Extensive Games with Perfect Information

An extensive game is a detailed description of the sequential structure of the decision problems encountered by the players in strategic situation. There is perfect information in such a game if each player, when making any decision, is perfectly informed of all the events that have previously occurred.

7.1.1 Definitions

Definition An **extensive game with perfect information** $\langle N, H, P, U_i \rangle$ has the following components:

- A set of N players
- A set H of sequences (finite or infinite). Each element of H is a **history**; each component of a history is an **action** taken by a player.
- P is the **player function**, $P(h)$ being the player who takes an action after the history h.
- Payoff function $U_i, i \in N$

After any history h player $P(h)$ chooses an action from the set $A(h) = \{a : (h, a) \in H\}$. The empty history is the starting point of the game.

Example

Two players want two identical objects. One of them propose an allocation which the other either accepts or rejects. Both players are reasonable.
In this representation each node corresponds to a history and any edge corresponds to an action.

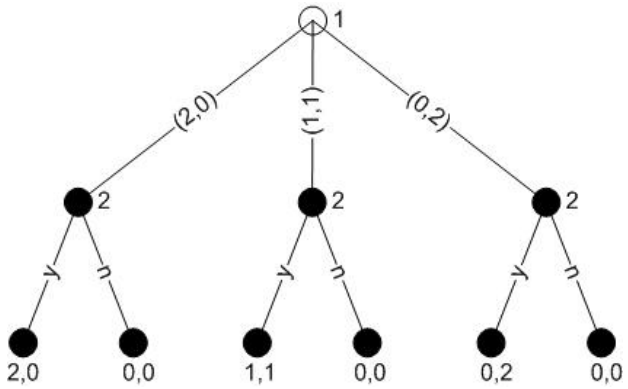


Figure 7.1: An extensive game, allocating two identical objects between two people

- $H = \{\emptyset, (2,0), (1,1), (0,2), ((0,2), y), ((2,0), n), ((1,1), y), ((1,1), n), ((0,2), y), ((0,2), n)\}$
- $P(\emptyset) = 1$ and $P((2,0)) = P((1,1)) = P((0,2)) = 2$

7.1.2 Strategy

Definition A **strategy of player** $i \in N$ in an extensive game $\langle N, H, P, U_i \rangle$ is a function that assigns an action in $A(h)$ to each history $h \in H$ for which $P(h) = i$

A strategy specifies the action chosen by a player for *every* history after which it is her turn to move, *even for histories that , if the strategy is followed, are never reached.*

Example

In the game shown in figure 7.2, the strategies of the first player $S_1 = \{AE, AF, BE, BF\}$, i.e. her strategy specifies an action after the history (A, C), even if she chooses B at the beginning of the game.
One can transform an extensive game with perfect information to a normal game by setting all the possible histories as the possible choices for a normal game.

7.1.3 Nash Equilibrium

Definition A **Nash equilibrium of an extensive game with perfect information** $\langle N, H, P, U_i \rangle$ is a strategy profile $s^* = (s_i)_{i \in N}$ such that for every player $i \in N$ and for every

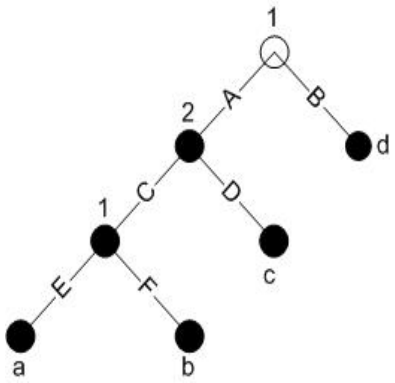


Figure 7.2: An extensive game in which player 1 moves before and after player 2

strategy s we have $U_i(s^*) \geq U_i(s)$

Example

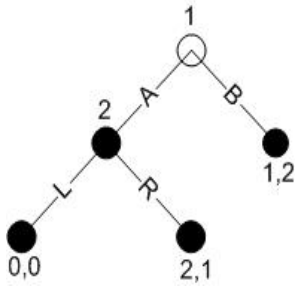


Figure 7.3: Two players extensive game

The game has two Nash equilibria: (A, R) and (B, L) with payoff $(2, 1)$ and $(1, 2)$. The strategy profile (B, L) is a Nash equilibrium because given that player 2 chooses L, it is optimal for player 1 to choose B at the beginning. (B, R) is not a Nash equilibrium since then player one prefer to choose A. Player 2's choice L is a "threat" if player 1 chooses A. If player 2 chooses R, then player 1 prefer A since her payoff increases.

7.1.4 Subgame perfect Equilibrium

Definition The **subgame of the extensive game with perfect information** $\Gamma =$

$\langle N, H, P, U_i \rangle$ that follows the history h is the extensive game $\Gamma(h) = \langle N, H|_h, P|_h, U_i|_h \rangle$ where $H|_h$ is the set of sequences $h\prime$ of actions for which $(h, h\prime) \in H$.

Definition A **subgame perfect equilibrium of an extensive game with perfect information** $\langle N, H, P, U_i \rangle$ is a strategy profile s^* such that for every player $i \in N$ and every history $h \in H$ for which $P(h) = i$ we have $U_i(s^*|_h) \geq U_i(s|_h)$ for every strategy s_i of player i in the subgame $\Gamma(h)$

Lemma 7.1 *The strategy profile s^* is a subgame perfect equilibrium if and only if for every player $i \in N$ and every history $h \in H$ for which $P(h) = i$ and for every $a_i \in A_i(h)$ exists $U_i(s^*|_h) \geq U_i(s^*_{-i}|_h, s_i)$ such that s_i differs from $s^*_i|_h$ only in the action a_i after the history h.*

Proof: If s^* is a subgame perfect equilibrium then it satisfies the condition. Now suppose there is a history h which player $P(h)$ should change her action. Let h be the longest history as above. For $P(h) = i$ she can change to $a_i \in A_i(h)$ and increases her payoff. Thus s^* is not a subgame perfect equilibrium. □

Theorem 7.2 *Every extensive game with perfect information has a subgame perfect equilibrium.*

Proof: We will use a backwards induction procedure. We start with the leaves and walk up through the tree. In every vertex we choose the best action (Best Response). By Lemma **??** this profile is a subgame perfect equilibrium. □

The Chain-Store Game

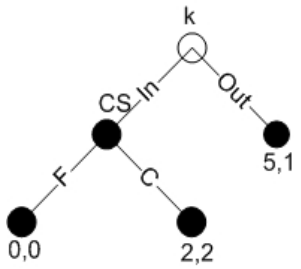


Figure 7.4: Player's choices in city k in the chain-store game

A chain-store (player CS) has branches in K cities. In each city k there is a single competitor, player k. In each period one of the potential competitors decides whether or

not to compete. If player k decides to compete then the chain-store can either fight (F) or cooperate (C). If challenged in any given city the chain-store prefers to cooperate rather then fight, but obtains the highest payoff if there is no entry. Each potential competitor is better off staying out than entering and being fought, but obtains the highest payoff when it enters and the chain-store is cooperative.
The game has a multitude of **Nash equilibria**: (Out, F) or (In, C).
The game has a unique **subgame perfect equilibrium**: every challenger chooses In and the chain-store always chooses C.

7.2 Repeated Games

The idea behind repeated games is that if we let the players play the same game a couple of times, they could get to different equilibria than those of a Nash Equilibrium of a one single round game. For example, we would like to achieve cooperation in the Prisoner's Dilemma game.

7.2.1 Finitely Repeated Games

Lets look again at the Prisoner's Dilemma game:

	C	D
C	$(3,3)$	$(0,4)$
D	$(4,0)$	$(1,1)$

Claim 7.3 *In a repeated game of T steps, where T is a final number, the only Nash Equilibrium is to play (D,D) in all T steps.*

Proof: In the last step, both players will play D, since otherwise at least one of the players would want to change her decision in order to improve her benefit. Now, using induction, if both players played the last i steps (D,D), then the same reason will hold for the $i-1$ step. □

We shall look now at a modified Prisoner's Dilemma game:

Claim 7.4 *In the finite T steps modified game, there is a subgame perfect equilibrium for which the outcome is (C,C) in every step but the last three, in which it is (D,D).*

Proof: The strategy chosen by the first player should be to play $T-3$ times C and then the last 3 times to play D. However, if the second player has played differently than this

	C	D	E
C	$(3,3)$	$(0,4)$	$(0,0)$
D	$(4,0)$	$(1,1)$	$(0,0)$
E	$(0,0)$	$(0,0)$	$(\frac{1}{2},\frac{1}{2})$

strategy, we will play E for the rest of the steps. Since we stop cooperating at the $T-2$ step, its enough to see if the other player has played differently at the $T-3$ step. Here are the two possible outcomes starting from the $T-3$ step:
Playing according to the strategy will yield (C,C) (D,D) (D,D) (D,D). The total payoff if these steps for the second player is $3+1+1+1=6$.
If the second player will change her strategy, the best moves that can be made are (C,D) (E,E) (E,E) (E,E). The total payoff for the second player is $4+\frac{1}{2}+\frac{1}{2}+\frac{1}{2}=5\frac{1}{2}$.
As we can see playing differently than the stated strategy will yield less profit for the deviating player. Thus it is best to play the proposed strategy by both players. □

The average payoff in this game is $(3(T-3)+3)/T$ which is $3-\frac{6}{T}$. This payoff is close to 3 which is the payoff of repeated cooperation.

7.2.2 Infinitely Repeated Games

There are several ways to look at the payoff of a player in an infinitely repeated game, a game that is repeated an infinite number of steps. We shall look at an N players game G with a payoff function $\vec{u}$, where u^i is the payoff function of player i. We define u_t^i as the payoff of player i at step t.

Definition The **average payoff** of a game G is the limit of the average payoff of the first T steps:
$\frac{1}{T}(\sum_{t=1}^{T} u_t^i) \rightarrow_{T\rightarrow\infty} \overline{u}^i$

Definition The **finite payoff** of a game G is the sum of the payoff of the first H steps of the game: $\sum_{t=1}^{H} u_t^i$

Definition The **discount payoff** of a game G is the weighted sum of the payoff of the steps of the game: $\sum_{t=1}^{\infty} u_t^i$

In the rest of the document when we refer to the payoff of an infinitely repeated game, we shall mean an average payoff $\overline{u}^i$.

Definition The **payoff profile** of an infinitely repeated game G is the payoff vector $\vec{w}$, where w_i is the payoff of player i. A payoff profile $\vec{w}$ is feasible if there are β_a for each

outcome $a \in A$, $K = \sum_{a \in A} \beta_a$, such that $\vec{w} = \sum_{a \in A} \frac{\beta_a}{K} \vec{u}(a)$.

Definition The **minimax payoff** of player i in a single step is: $v_i = \min_{a_{-i} \in A_{-i}} \max_{a_i \in A_i} u^i(a_{-i}, a_i)$

Claim 7.5 *In every Nash Equilibrium of a single game, the payoff of player i is at least v_i.*

Proof: If a player has a smaller payoff than v_i then by the definition of the minimax payment, there is a different strategy that she can play in order to profit at least v_i. □

Definition A payoff profile $\vec{w}$ is **enforceable** if $\forall_{i \in N} v_i \leq w_i$. A payoff profile is **strictly enforceable** if $\forall_{i \in N} v_i < w_i$.

Theorem 7.6 *Every feasible enforceable payoff profile $\vec{w}$ in an infinitely repeated game G is a Nash Equilibrium with an average payoff.*

Proof: We will describe a strategy that is Nash Equilibrium with the payoff $\vec{w}$.
Since $\vec{w}$ is feasible there are β_a for each $a \in A$, $K = \sum_{a \in A} \beta_a$, such that $\vec{w} = \sum_{a \in A} \frac{\beta_a}{K} \vec{u}(a)$.
We shall assume that $\forall_{a \in A} \beta_a \in \mathbf{N}$.
The strategy of each player is to play cycles of K steps, going over all the possible outcomes $a \in A$ in an ordered list and playing her outcome in a β_a times. If player i deviates from this strategy, the rest of the players will change to a new strategy P_{-i} that enforces the payoff of player i to be at most the minimax payoff v_i.
Thus, if a player i deviates from the main strategy, her payoff will be v_i. which is not better than her payoff in $\vec{w}$. Because each deviation will not improve the payoff of player i, $\vec{w}$ is a Nash Equilibrium. □

Theorem 7.7 *Every feasible strictly enforceable payoff profile $\vec{w}$ in an infinitely repeated game G has a Subgame Perfect Equilibrium with an average payoff.*

Proof: We will describe a strategy that is Subgame Perfect Equilibrium with the payoff $\vec{w}$.
We shall use the same cyclic strategy as in the previous theorem, where all the players play the outcome a for β_a steps. If a player deviates from the strategy, the other players will punish her but only in a finite number of steps. At the end of the punishing steps all players will resume to play the cyclic strategy.
More specifically, at the end of each K steps cycle, the players will check if one of the players has deviated from the cyclic strategy. If a player, let say player j, has indeed played differently, the other players will play, for m^* steps, the minimax strategy P_{-j} that will enforce a payoff of at most v_j for player j, where m^* is chosen as follows:
We mark player j's strategy in each step t of the last cycle as $a_j'^t$. The maximal payoff benefit

for player j out of the steps in the cycle is $g^* = \max_{a^t}[u^j(a^t_{-j}, a'^t_j) - u^j(a^t)]$, where $a^t \in A$ is the expected outcome in step t of the K steps. We would like to get $Kg^* + m^*v_j < m^*w_j$ in order to make the punishment worthwhile. However, since $\vec{w}$ is strictly enforceable, we know that $v_j < w_j$ and so there exist m^* such that $0 < \frac{Kg^*}{w_j - v_j} < m^*$.
Playing the punishment strategy for m^* steps will yield a strictly smaller payoff for player j than playing the cyclic strategy without deviation. □

7.3 Bounded Rationality

We have seen that in a repeated Prisoner's Dilemma game of N rounds there is no cooperation. A way to circumvent this problem is to assume that the players have limited resources. This kind of player is said to have **Bounded Rationality**.

A Bounded Rationality player is an automata with:

S - the state set.

A - the actions.

$\delta : S \times A \rightarrow S$ - the state transfer function.

$f : S \rightarrow A$ - the actions function.

S_0 - the starting state.

We assume that the automata is deterministic and that in each state only one action is chosen. A stochastic strategy is to randomly choose one automata from a set of deterministic automatas.

7.3.1 Tit for Tat

The Tit for Tat strategy (TfT) for the repeated Prisoner's Dilemma game will play in the next round what the opponent played in the last round (see figure **??**).

Theorem 7.8 *The TfT strategy is a Nash Equilibrium if the opponents has at most $N - 1$ states while the game is of N rounds.*

Proof: Any diversion of the opponent from the cooperation action C for $k > 0$ rounds, playing D, will yield a profit of $4 + 1(k - 1) + 0\delta$ for the opponent. On the other hand, if a cooperation is kept, the opponent's profit is $3k + 3\delta$. Where δ is 1 when the diversion is not at the end of the game (i.e. there are more rounds afterwards) and 0 at the end of the game. If δ is 1, the next round the opponent will play C, she will profit 0, while profiting 3

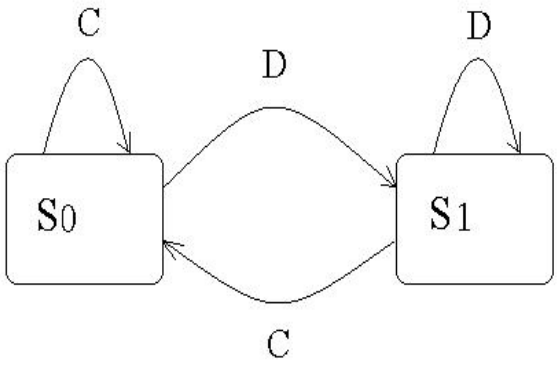


Figure 7.5: The automata of the Tit for Tat strategy

if a cooperation was maintained. This means that in the middle of the game the opponent will always prefer to cooperate since it has a bigger profit, $3k+3 > 3+k$. The problem is at the end of the game. While $k > 1$ it is still more profitable to cooperate, since $3k > 3+k$. However when k is 1, it is better to defect, meaning that the only time that it is better to defect is in the last round.

If the opponent has an automata of at most $N-1$ states then for any action series of length $N-1$ the automata will return to an already visited state, arriving at a cycle. If until then the automata did not play D, it will never play D. However, if it did play D in one of the first $N-1$ rounds the opponent will gain less than playing a full cooperation. Thus, the only Nash Equilibrium for at most $N-1$ states automata is to play C constantly.

Since this logic is true for any kind of at most $N-1$ states automata, it is also true for a stochastic strategy over a set of such automatas. □

7.3.2 Bounded Prisoner's Dilemma

We have seen an example for a simple bounded rationality strategy for the Prisoner's Dilemma game that will yield a cooperation in some conditions. The next step is to analyze any general bounded rationality strategy for that game, described by a final automata, and find what are the conditions that will lead to a cooperation between two players using these strategies in an N rounds game.

Theorem 7.9 *If, in a Repeated Prisoner's Dilemma with N rounds, both players have an automata with at least 2^{N-1} states then the only Equilibrium is the one is which both players play (D,D) in all rounds.*

Proof: Given an opponent, that optionally can play stochastically, it is possible to play an optimal strategy as follows:

We shall build a game history tree of depth N. At each node we shall calculate the distribution of the optimal action using dynamic programming, starting from the leafs and up the tree. Based on the profits of all the possible paths from a specific node to the leafs, we can choose the best response at every node.

The chosen optimal startegy can be encoded in a full binary tree of depth $N-1$, describing the first $N-1$ rounds, and one node for the last round (any optimal automata plays D in the last stage for every history of nonzero probability), summing to 2^{N-1} states.

As this is the optimal unrestricted strategy, the only Equilibrium is to play (D,D) at each round, as shown earlier. □

Theorem 7.10 *For a Repeated Prisoner's Dilemma with N rounds, when both players have automatas with at most $2^{\epsilon_1 N}$ states (when it is possible to change to a different automata with a related size boundary $2^{\epsilon_2 N}$), there exists an Equilibrium with a profit of $3-\epsilon_3$.*

Computational Game Theory **Spring Semester, 2003/4**

Lecture 8: May 11, 2004

Lecturer: Yishay Mansour *Scribe: Eitan Yaffe, Noa Bar-Yosef*

8.1 Regret

Our goal is to build a strategy with good performance when dealing with repeated games. Let us start with a simple model of regret.

8.2 Basic Model

Assuming that the opponent has the same stochastic strategy at each step, how should we play?
Let's formalize this:

- N actions
- For each step t, we choose a distribution p^t over the N actions
- For each step, we have a loss l^t where $l^t(i) \in [0,1]$ is the loss from action i
- Our loss is $\sum_{i=1}^{N} p^t(i) l^t(i)$

Note that we do not rely on the number of opponents or on their actions. Once we assume that the opponent is constant and does not depend on us, then the opponent's influence is only on the cost of each action.
Our goals:

- Bring the loss to a minimum
- Choose the best action (the opponent does not change throughout time)

8.3 A Greedy Algorithm

One way to implement our goal is by using the greedy algorithm:

- For the $t+1$ step, we will calculate for each i:

$$L_i^t = \sum_{k=1}^{t} l^k(i) = L_i^{t-1} + l^t(i)$$

- For the $t+1$ step, we will chose the best action that we had up until now:

$$a^{t+1} = arg\min_i L_i^t$$

We can see that this is not the optimal algorithm, because although the history gives us a good hint regarding the probability of each action, this does not necessarily give the best action, but rather an approximation of the best action.
To simplify, let's assume that $l^t(i) \in \{0,1\}$ and define $p_i = Pr[l^t(i) = 1]$ (the probability of the loss of action i to be 1). The best action is $a^* = arg\min_i p_i$. Thus the average loss is p^* for each step. And so we get that the optimal loss for T steps is p^*T.
The loss of the Greedy algorithm is $L_G^T = \sum_{k=1}^{T} Greedy^k$. Define R (The regret) to be: $R = L_G^T - p^*T$. We get that the average of R is:

$$E[R] = \sum_{k=1}^{T} E[Greedy^k] - p^*$$

We analyze this by looking at each step k, separately. When k is large enough, it converges to p^*.
The average we get for the i-th action is: $\hat{p}_i^k = \frac{L_i^k}{k}$.
Thus, using the Chernoff bound, we get:

$$Pr[|p_i - \hat{p}_i^k| \geq \epsilon] \leq e^{-2\epsilon^2 k}$$

If for all the actions $Pr[|p_i - \hat{p}_i^k|] \leq \epsilon$, then:

$$E[Greedy^k] - p^* \leq 2\epsilon + Ne^{-2\epsilon^2 k}$$

Taking $\epsilon = \sqrt{\frac{\ln Nk}{k}}$, results in: $O(\sqrt{\frac{\ln NT}{k}})$
Now we just need to sum up all the losses:

$$\sum_{k=1}^{T} \sqrt{\frac{\ln NT}{k}} \approx \sqrt{\ln NT} \int_1^T \frac{1}{\sqrt{k}} dk = O(\sqrt{T \ln NT})$$

Meaning that the extra loss (the regret) is:

$$E[R] = O(\sqrt{T \ln NT})$$

Note that this bound is not tight.

8.4 External Regret

In the above analysis we assumed that the system does not change throughout time. We would like to change this assumption, but this means we must change our analysis (e.g. we cannot use p^* since it changes over time and isn't defined). First we shall consider comparing to the performance of OPT. This turns out to be a not a very informative measure. Then we shall introduce the *External Regret* measure. Consider the following example:

8.4.1 "Bad" example

On each step OPT chooses a random $i \in N$ such that: $l^t_{j \neq i}(j) = 1$ and $l^t(i) = 0$. We can see that the average loss for any online algorithm is at least $1 - \frac{1}{N}$ (on average), while OPT's loss is 0.

8.4.2 Definition

Instead of comparing to OPT we can compare our performance to the performance of the single best action: $L_{min} = min_i L^T_i$. In general, for any Algorithm (or Hypothesis) denoted H, we define its loss on step t to be L^t_H and its overall loss to be $L^T_H = \sum_{t=1}^{T} L^t_H$. The *External Regret* is defined as follows:

$$R^{ext} = L^T_H - L_{min}$$

8.4.3 Analyzing the greedy algorithm

The loss of the Greedy algorithm over T steps is L^T_G. Reducing R^{ext} means coming as close as possible to L_{min}.

Claim 8.1 $L^T_G \leq N(L_{min} + 1)$

Proof: For simplicity we assume that $l^t(i) \in \{0, 1\}$. At step k, let $b_k = \min_i L^k_i$ and $n_k = |\{i : L^k_i = b_k\}|$. We define a lexicographic order over the pairs (b_k, n_k). On each step that *Greedy* losses 1, either b_k increases by 1 or n_k decreases by 1. Note that n_k can decrease only $N - 1$ times, consecutively. Therefore the loss of *Greedy* is bounded by $b_k N$. □

The problem of deterministic algorithms is that we can always create a large loss. The following (bad) example, is true for every deterministic algorithm.

8.4.4 Example

At the k-th step, the opponent's algorithm chooses action a_k. The loss is $l^k(a_k) = 1$ for action k and for the rest of the actions it is 0. The loss of the online deterministic algorithm is thus T, while there exists an action whose loss is $\frac{T}{N}$. (Since the sums of 1's is T, then the average is $\frac{T}{N}$).

8.4.5 A stochastic strategy

We will examine a strategy that has an (expected) external regret of $O(\sqrt{L_{\max} \log N} + \log N)$. What we shall do is build a distribution over the actions, dependant on the Regret.
We define l_H^t as the loss of the online algorithm in the t-th step: $l_H^t = \sum_{i=1}^{N} p^t(i) l^t(i)$ and $R_a^T = \sum_{t=1}^{T} [l_H^t - l^t(a)]$. We also define a "pseudo-regret" where we multiply our loss by $0 < \beta < 1$:

$$\tilde{R_a^T} = \sum_{t=1}^{T} [\beta l_H^t - l^t(a)]$$

It is easy to see that $R_a - \tilde{R}_a$ is small for $\beta \approx 1$. We now build the exponential weights that are dependant on $\tilde{R}_a$:

$$w_a^0 = 1$$
$$w_a^{t+1} = w_a^t \beta^{l^t(a) - \beta l_H^t}$$

According to these weights, we can define the probabilities by normalizing:

$$W^{t+1} = \sum_{a \in N} w_a^{t+1}$$

$$p_a^{t+1} = \frac{w_a^{t+1}}{W^{t+1}}$$

Claim 8.2 $0 \leq \sum_{a \in N} w_a^t \leq N$

In other words, all the weights are positive and do not "run off" to very large sizes.
The proof of this claim will appear shortly, but in the meanwhile until then we will assume its correctness.
Using the claim, we get that for each action a:

$$w_a^T = \beta^{L_a^T - \beta L_H^T} = \beta^{-\tilde{R}} \leq N$$

Comparing $\beta^{L_a^T - \beta L_H^T} \leq N$, and taking ln, results in:

$$(L_a^T - \beta L_H^T) \ln \beta \leq \ln N$$

Dividing by $\ln(\beta)$:

$$L_a^T - \beta L_H^T \geq -\frac{\ln N}{\ln \frac{1}{\beta}}$$

$$\frac{L_a^T}{\beta} + \frac{\ln N}{\beta \ln \frac{1}{\beta}} \geq L_H^T$$

Choosing $\beta = 1 - \gamma$, $ln\frac{1}{\beta} \approx \gamma$,

$$L_a^T + \frac{\gamma}{1-\gamma} L_a^T + 2\frac{\ln N}{\gamma} \geq L_H^T$$

Note that $\frac{\gamma}{1-\gamma} L_a^T + 2\frac{\ln N}{\gamma}$ is basically the Regret. Defining $L_{\max} = \max_a L_a^T$ we have:

$$\gamma L_{\max} = \frac{\ln N}{\gamma}$$

$$\gamma = \sqrt{\frac{\ln N}{L_{\max}}} < \frac{1}{2}$$

$$\Rightarrow R = O(\sqrt{\ln N \cdot L_{\max}} + \log N)$$

In each step, we incur a loss of at most 1, thus $L_{\max} \leq T$.
In each step, the opponent chooses some kind of loss that can be dependant on our distribution. Nevertheless, we are able to approach quite well, as we can see:

$$L_H^T \leq L_{\min} + O(\sqrt{T \ln N})$$

where $L_{\min} = \min_a L_a^T$.

We shall now proceed to prove our above claim about our weights that states that:

$$0 \leq \sum_{a \in N} w_a^t \leq N$$

Proof: Trivially, $0 \leq \sum_{a \in N} w_a^t$.
We are left to see that the weights are in fact bounded:

$$l_H^t = \sum_{a \in N} P^t(a) l^t(a) = \sum_{a \in N} \frac{w_a^t}{W^t} l^t(a)$$

$$W^t l_H^t = \sum_{a \in N} w_a^t l^t(a) \tag{8.1}$$

We can give a linear bound for the function β^x (relying on it's convexity) for any $\beta \in [0,1]$. For $x \in [0,1]$ we know that $\beta^x \leq 1-(1-\beta)x$. For $x \in [-1,0]$ we get $\beta^x \leq 1 - \frac{1-\beta}{\beta}|x|$.
Now, by the definition of w_a^{t+1}:

$$\sum_{a \in N} w_a^{t+1} = \sum_{a \in N} w_a^t \cdot \beta^{l^t(a) - \beta l_H^t}$$

By the above properties of beta we get the bound:

$$\leq \sum_{a \in N} w_a^t (1-(1-\beta)l^t(a))(1+(1-\beta)l_H^t)$$

By opening the parenthesis and discarding $(1-\beta)l^t(a)(1-\beta)l_H^t$ since it is negative (the product of positive and negative)

$$\leq \sum_{a \in N} w_a^t - (1-\beta)[\sum_{a \in N} w_a^t \cdot l^t(a)] + (1-\beta)[\sum_{a \in N} w_a^t \cdot l_H^t]$$

Using equation (8.1):

$$\leq \sum_{a \in N} w_a^t \leq N$$

□

8.4.6 Example

We have shown a probabilistic algorithm whose bound is relatively tight. An example for the tightness of its bound follows:
Let the loss of one action be $\frac{1}{2} - \epsilon$ and the loss of the rest of the actions be $\frac{1}{2}$. Taking $\epsilon = \frac{1}{\sqrt{T}}$ and randomly choosing between these two actions, will not be able to incur a loss of less than $\sqrt{T}$ the overall loss.

Up until now we've discussed **External Regret** whereas:

$$R^{ext} = L_H^T - L_{\min}$$

i.e. Our loss is not measured according to our algorithm, but rather in relation to each separate action taken. We can easily define other measures...

8.5 Correlated Equilibrium

- Game with M players
- A_i - N actions of player i
- S^i - The loss function of player i:

$$S^i : A_i \times (\times A_j) \to [0, 1]$$

Definition L et Q be a distribution over the joint actions $(\times A_i)$, such that for each player i and for each action $\alpha \in A_i$:

$$E_{a\sim Q}[S^i(a_i, a^{-i})|a_i = \alpha] \leq E[S^i(b, a^{-i})|a_i = \alpha]$$

In other words, this means that given an action a_i from the distribution Q to player i, then this is also his *best response* to play it!
We can formalize this also in a different manner:
Let us define the function $F : A_i \to A_i$, then Q is a *Correlated Equilibrium* if for each player i and for each F we have:

$$E_{a\sim Q}[S^i(a_i, a^{-i})] \leq E_{a\sim Q}[S^i(F(a_i), a^{-i})]$$

Furthermore we will now define that Q is *Epsilon-Correlated* when for each player i and for each F:

$$E_{a\sim Q}[S^i(a_i, a^{-i})] \leq E_{a\sim Q}[S^i(F(a_i), a^{-i})] + \epsilon$$

This means that F "exchanges" the actions of i according to the suggestions of Q. We bound the gain from using F by ϵ.

We define the *Swap Regret* to be:

$$R^{swap} = \max_F\{L_H - L_{H,F}\} = \sum_{t=1}^{T} p^t(i)[l^t(i) - l^t(F(i))]$$

Claim 8.3 *Let us assume a game with M players where each player plays a strategy that has $R^{swap} \leq R$. Let p be the empirical distribution at time $[1, T]$ (This means that each step has a vector of actions and for each one of these vectors we will give a probability). Then:*

1. *The average loss of the player according to p is his loss in the "game".*
2. *p is ϵ-correlated for $\epsilon = \frac{R}{T}$. This is true because every player can gain no more than R by using F.*

8.5.1 Internal Regret

We also define *Internal Regret*: For $a_i, a_j \in A$, we swap $a_i \rightarrow a_j$. (We swap only between a pair of actions).

8.5.2 Reduction of External Regret to Swap Regret

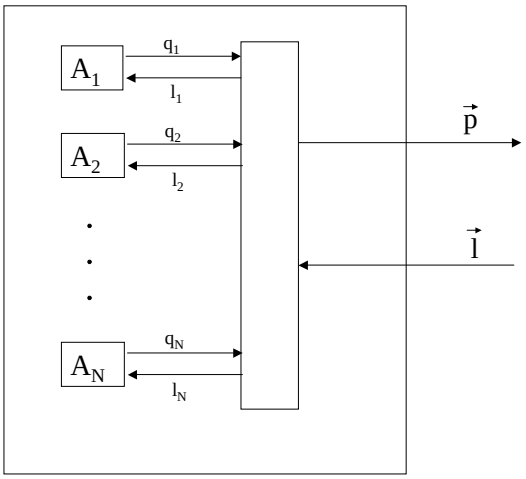


Figure 8.1: Reduction of External Regret to Swap Regret algorithm

Having already discussed R^{ext}, we now present a reduction from R^{ext} to R^{sw}:
For each action i, there will be an algorithm A_i. Intuitively, the goal of A_i is to avoid regret by replacing action i with any other action. We construct an algorithm which combines N algorithms, as shown in figure 8.1. Each algorithm guarantees us of a small R^{ext}. Each algorithm outputs a vector of what it would like to play, and we need to return to each separate algorithm its loss. We need to wrap up these algorithms in some sort of interface which will calculate the distribution and return the loss. Thus we have two important actions to do:

1. Calculate p^t from $\vec{q_1^t}, ..., \vec{q_N^t}$
   For this we will choose a distribution p such that: $p = p \cdot Q$ where $\vec{q_i}$ is the i-th row of Q. Specifically:

$$\forall j \; p_j^t = \sum_{i=1}^{N} p_i^t \cdot q_{i,j}^t$$

   This means that choosing an action j according to p is equivalent to choosing an algorithm A_i according to p, and then choosing action j according to A_i.

2. "Distribute" the loss of $\vec{l^t}$ to $\vec{l_1^t}, ..., \vec{l_N^t}$.
   Upon receiving $\vec{l^t}$, we return $p_i^t \cdot \vec{l^t}$ to A_i. The loss that A_i "sees" is:

$$(p_i^t \cdot \vec{l^t})\vec{q_i^t} = p_i^t(\vec{q_i^t} \cdot \vec{l^t})$$

   Thus, for each A_i and for each action j we have a bound on the regret:

$$\sum_{t=1}^{T} p_i^t(\vec{q_i^t} \cdot \vec{l^t}) \leq \sum_{t=1}^{T} p_i^t \cdot l^t(j) + R_i$$

   (R_i may be dependant on T, N, or a loss as before, but it is not dependant on the game itself.)
   When we sum up the losses, we get that for any point in time:

$$\sum_{i=1}^{N} p_i^t(\vec{q_i^t} \cdot \vec{l^t}) = p^t \cdot Q \cdot l^t = p^t \cdot l^t = l_H^t$$

   Therefore we get in total:

$$\sum_{i=1}^{N}[\sum_{t=1}^{T} p_i^t(\vec{q_i^t} \cdot \vec{l^t})] = \sum_{t=1}^{T} l_H^t = L_H^T \leq \sum_{i=1}^{N}\sum_{t=1}^{T} p_i^t \cdot l^t(F(i)) + \sum_{i=1}^{N} R_i$$

   However, $\sum_{i=1}^{N}\sum_{t=1}^{T} p_i^t \cdot l^t(F(i)) = L_{H,F}^T$ and so this results in:

$$L_H^T \leq L_{H,F}^T + \sum_{i=1}^{N} R_i$$

   Recall that we previously proved that: $R = O(\sqrt{T \log N} + \log N)$ so by summing over all R_i we have that:

$$R^{sw} = O(N\sqrt{T \log N} + N \log N)$$

   Thus,

$$R = O(\sqrt{L_{\max_i} \log N} + \log N)$$

   And finally,

$$L_H \leq L_{H,F}^T + O(\sqrt{L_{\max_i} \log N} + \log N)$$

   Since in our case, $\sum L_{max_i} \leq T$, the worst case is when $L_{max_i} = \frac{T}{N}$.

Prior knowledge of L_{max}

We need to know L_{max_i} in order to define A_i. We will change our previous algorithm for External Regret so that it won't need to have L_{max} beforehand.
We start with $L_{max} = 1$, and each time we reach the bound, we multiply L_{max} by 2 and start over again.
We now have that the regret is:

$$\sum_{j=1}^{\log L_{\max}} O(\sqrt{2^j \log N} + \log N) = O(\sqrt{L_{\max} \log N} + \log T \log N)$$

It is easy to see that our new bound is a bit worse than the previous, but here we do not need to rely on knowing $L_{\max}$.
Using the new algorithms, we get that the worst case is (still): $L_{\max_i} = \frac{T}{N}$ and thus:

$$L_H \leq L_{H,F} + O(TN \log N + N \log N \log T)$$

Computational Game Theory **Spring Semester, 2003/4**

Lecture 9: Dynamics in Load Balancing

Lecturer: Yishay Mansour *Scribe: Anat Axelrod, Eran Werner*

9.1 Lecture Overview

In this lecture we consider dynamics of a load balancing system. We study the number of steps required to reach pure Nash equilibrium in a system of "selfish" and greedy players (jobs). We are interested in the convergence time to Nash equilibrium, rather than the quality of the allocation. We consider a variety of load balancing models including identical, restricted, related and unrelated machines. We compare different scheduling strategies such as allowing the largest job to move first (or smallest weight first) and compare their effect on the convergence time. Note that the discussion can be easily extended to the domain of routing problems.
The lecture is divided into two major sections. In the first we deal with migration policies that allow a single user to migrate at a time. In the second part we discuss the case where users migrate concurrently.
For the former part we consider two settings:

1. An empty system where jobs are added gradually. We will show an upper bound of $O(n)$ for this case using the Max Weight First policy.

2. A system at an arbitrary initial state. Here we begin by showing that any scheduler will converge, even in the unrelated and restricted assignment model. Afterwards, we give special treatment to the case of identical machines where we show both an upper bound of $O(n)$ using Max Weight First and a lower bound of $\Omega(n^2)$, on the other hand, using Min Weight First.

When users are allowed to migrate concurrently from overloaded to underloaded machines, we present two algorithms for the special setting of two identical machines and we conclude by showing a general algorithm for multiple identical machines.

9.2 The Model - Formal Definition

In our load balancing scenario there are m parallel machines and n independent jobs. At each step each job is assigned to exactly one machine.

- S_i - The speed of machine M_i. $S_i \in [1, S_{max}]$ where the speeds are normalized s.t. $S_{min} = 1$ and the maximum speed is S_{max}. When considering the identical or unrelated machines models, $S_i = 1$ for $1 \leq i \leq m$.

- W_j - The weight of job j. We will mainly deal with the related and identical machine models. When generalizing to the unrelated machine models we denote $w_{i,j}$ as the weight of job j when assigned on machine M_i.

- $B_i(t)$ - The set of jobs on machine M_i at time t.

- $L_i(t)$ - The load of machine M_i at time t. This is the sum of the weights of the jobs that chose M_i. $L_i(t) = \sum_{j \in B_i(t)} W_j$.

- $T_i(t)$ - The normalized load on machine M_i at time t, obtained by dividing the load of the machine with its speed. $T_i(t) = \frac{L_i(t)}{S_i}$

- $U(t)$ - The set of jobs that may decrease their experienced load at time t by migrating to another machine. A job wishes to migrate if and only if its cost (load) will strictly reduce after the migration. Note that the users (jobs) are considered to be myopic i.e. at each step they wish only to improve their state regardless of future consequences. We examine arbitrary improvement steps, and when the job selects a machine which minimizes its cost it is said to have played the best response policy.

- ***The Scheduler*** - Before migrating between machines a job needs to receive a grant from the centralized scheduler. The scheduler does not influence the selection of the target machine but merely controls the order in which the jobs migrate.

- ***Scheduling Strategies*** - The input at time t is the set of jobs $U(t)$ and the output is a job $j \in U(t)$ which is allowed to migrate at time t. When only one job is allowed to migrate at a time it is easy to define some natural selecting strategies, for example:

  - Max Weight First
  - Min Weight First
  - Arbitrary Job Next

- ***Pure Nash Equilibrium*** - A state in which no job can benefit from unilaterally migrating to another machine. For every user j on machine M_i it holds that $\forall k \;\; T_i \leq T_k + w_{k,j}$.

9.2.1 Dynamics of Migration

We consider two models in which jobs are allowed to migrate in order to improve their costs. First, we examine a scenario where at each step a single user (job), selected by the scheduler from the group of "unsatisfied" jobs, is allowed to move. In the latter model this group of jobs is allowed to migrate simultaneously.

9.3 Single Migrator

We will divide our discussion into two settings:
In 9.3.1 we start from an empty system and gradually add one job at a time.
In 9.3.2 we consider starting from an arbitrary state, then applying the scheduling policy until the system converges to equilibrium.

9.3.1 Starting from Scratch

Let the jobs enter the system in descending sorted order of weights:
$w_1 \geq w_2 \geq ... \geq w_n$. When entering the system, each job immediately selects its best response.

Claim 9.1 *The system will remain in Nash equilibrium at each step.*

Proof. By induction on the number of steps.
Trivially, the system is in Nash for the initial step of an empty system.
Assume that it is also in Nash at step $j-1$, immediately before w_j is added.
w.l.o.g job j selects machine M_1.
Obviously, all jobs on machines other than M_1 remain in equilibrium. It is enough to show that all jobs on M_1 are in equilibrium as well. Assume by contradiction that there exists a job k on M_1 which wishes to migrate to machine M_2. Then, $T_1 > T_2 + \frac{w_k}{S_2}$. Since $w_j \leq w_k$ it is also true that $T_1 > T_2 + \frac{w_j}{S_2}$, contradicting our assumption that machine M_1 was the best response for (and selected by) job j. □

Food For Thought 1 *How would you show an existence of a pure Nash equilibrium in the unrelated machines model?*

9.3.2 Starting from Something

Now the system is initially in an arbitrary state and we seek to reach equilibrium by a sequence of steps in which only one job, selected by the scheduler, can move. This model is called **E**lementary **S**tepwise **S**ystem (ESS).

Unrelated Machines

First we will show that even for the general model of unrelated machines with restricted assignments, convergence to equilibrium is promised. Furthermore, we establish this result without relying on best response moves but only general improvement steps. We obtain this result by identifying a potential function and showing that the potential strictly decreases after each improvement step. Consider the sorted vector of machine loads. We show that defining a lexicographic order on those sorted vectors provides us with a potential function. Using lexicographic order, vectors are compared by their first unequal component. One vector is called lexicographically larger than the other if its first non equal component is larger than its corresponding counterpart in the second vector.

Claim 9.2 *The sorted lexicographic order of the machine loads decreases when a job migrates via improvement.*

Proof. Improvement influences only two components of the sorted vector: one corresponding to the load on the machine that the job has left while the other to the machine the job has joined. The load on the machine the job joined cannot be higher than the load it experienced on the machine it had left (otherwise would it have moved?). Formally, if the job migrated from machine M_i to machine M_j, we have $L_i(t) > L_j(t+1)$. Additionally, $L_i(t) > L_i(t+1)$ since the job had left. Thus, if $L_i(t)$ was the k^{th} component in the sorted vector of loads then the respective component in the new vector cannot be larger and the new vector is considered lexicographically smaller. □

Since we have shown that improvement always decreases the potential, this gives us an upper bound on the convergence time which is equal to the number of different sorted loads vectors (trivially bounded by the number of configurations) - m^n.

Corollary 9.3 *For any ESS strategy, the system of unrelated machines with restricted assignments reaches Nash equilibrium in at most m^n steps.*

This result was obtained for any scheduler. We now wish to investigate how specific scheduling strategies may give better bounds. Unfortunately, we can provide results only in more specific models.

Identical Machines - Upper Bound

In the identical machines model with unrestricted assignments we use the Max Weight First (MWF) scheduling strategy to establish an upper bound of O(n) on the convergence time. The scheduler always selects a job $J(t) = \arg\max_{j \in U(t)}\{w_j\}$, i.e. the largest unsatisfied job. We assume that the jobs play their best response.

Claim 9.4 *Suppose that a job J has migrated to machine M which is its best response at time t_0. If J wishes to migrate again at a later time $t_2 > t_0$ then another job J' such that $w_{J'} > w_J$ must have joined machine M at time t_1, $t_0 < t_1 \leq t_2$.*

Proof. Since J wishes to migrate, one of two things must have happened at some earlier time t_1:

1. Job J' has moved from machine $M_{J'}$ to a different machine $M' \neq M$.
2. Job J' has joined job J at machine M.

1. Obviously, Job J does not wish to deviate to M' since it did not wish to do so prior to the move of J', and now the load on M' has only increased.
   Fortunately, job J has also no desire to move to $M_{J'}$ even though its load decreased. The reason for this is:
   $T_{M_{J'}}(t_1 - 1) > T_{M'}(t_1 - 1) + w_{J'} = T_{M'}(t_1)$ (J' shifted from $M_{J'}$ to M').
   ⇓

$T_{M_{J'}}(t_1) = T_{M_{J'}}(t_1 - 1) - w_{J'} > T_{M'}(t_1 - 1)$.
The resulting load on $M_{J'}$ is higher than the load that was on M' at time $t_1 - 1$. Since M' was not a best response for J at that time $M_{J'}$ certainly isn't now.

2. In this case we want to show that J will want to leave M only if $w_{J'} > w_J$. Assume by contradiction that $w_{J'} \leq w_J$. Let M' be the machine that J wishes to move to. Thus, $T_M(t_1) = T_M(t_1 - 1) + w_{J'} > T_{M'}(t_1 - 1) + w_J \geq T_{M'}(t_1 - 1) + w_{J'}$ Hence, $T_M(t_1) \geq T_{M'}(t_1 - 1) + w_{J'}$ contradicting the assumption that J' played his best response at time t_1.

□

Theorem 9.5 *The Max Weight Job strategy with best response policy, for a system of identical machines with unrestricted assignments, reaches Nash equilibrium in at most n steps.*

Proof. By claim 9.4 once a job has migrated to a new machine, it will never leave it unless a larger job arrives. Since the jobs arrive in descending weight order (MWF) only smaller jobs may arrive at subsequent steps. Therefore each job stabilizes after its first migration, and the theorem follows. □

Identical Machines - Lower Bound

We now present a lower bound using the Min Weight strategy. We demonstrate a setting with two identical machines and n jobs. Later, we explain the idea for generalizing the result to m machines.
Consider the following scenario. There are $\frac{n}{2}$ classes of jobs $C_1, ..., C_{\frac{n}{2}}$. Each class C_k contains exactly two jobs with weights $w_k = 3^{k-1}$. Notice that a job in C_k has weight equal to the total weight of all of the jobs in the first $k-1$ classes plus 1. Initially all jobs are located at the first machine. The scheduling process is divided into phases. At each phase k all jobs from classes C_1 to C_k except one job from C_k move from one machine to another. Thus, the duration of phase k is $2k - 1$. The scheduling consists of phases $\frac{n}{2}, ..., 1$, since after each phase the two jobs of the heaviest class are set on two different machines and the scheduling can be regarded as continuing recursively with one less class. At equilibrium each machine will occupy exactly one job from each class. The convergence time is given by the recursive formula:

$$f(r) = f(r-1) + 2r - 1$$

for $r = \frac{n}{2}$ and this is clearly $\Omega(n^2)$.
For the general lower bound we have $m = k + 1$ machines and n jobs. We divide the jobs into k weight classes of size $\frac{n}{k}$. The weight of a job in class j is chosen such that it is larger than the sum of all the job weights of earlier classes: if we denote by w_j the weight of a job $\in C_j$ then, $w_j > \sum_{i=1}^{j-1} \frac{n}{k} w_i$. Initially all jobs from class i are assigned to machine i and machine 0 is empty. Applying the Min weight scheduling consists of phases such that before a job from class k is granted to move all the lighter

jobs are equally distributed between machines $[0, k-1]$. However, as soon as the first heavy job is allowed to move it causes all of the lighter jobs to shift among the other machines, not containing a heavy weight job, and this process continues recursively, as the scheduling is from the lightest job first. This example gives a lower bound of $\Omega(\frac{(\frac{n}{k})^k}{2(k!)})$

Food For Thought 2 *Could you think of a natural scheduling strategy that would quickly converge to equilibrium in the related (or maybe even unrelated) model ?*

9.4 Concurrent Jobs Migration

In this section we examine the setting of n identical users i.e. $\forall j \quad w_j = 1$. For simplicity we also deal with identical machines i.e. $\forall i \quad S_i = 1$. With contrast to the previous model, several users can now move concurrently. Consequently, equilibrium is no longer promised, as the system may oscillate. For example, consider a two machine setting, where all jobs are initially placed on one of the machines. Since they all want to migrate at once they will find themselves all in a similar situation but on the second machine. We overcome this problem by introducing randomization into the decision if a job should migrate.
The system is at Nash equilibrium at time t if for every pair of machines M_i, M_j it holds that $L_i(t) \leq L_j(t) + 1$.

We will divide our discussion into two:
In 9.4.1 we start from the specific case of two identical machines.
In 9.4.2 we extend our discussion and deal with multiple identical machines.

9.4.1 Two Identical Machines

The Balance Algorithm

First we consider the case of only two identical machines M_1, M_2 with load functions $L_1(t), L_2(t)$ respectively. We use two variables $over, under \in \{M_1, M_2\}$ to identify at each step which is the more/less loaded machine i.e. its load is above/under the average load ($\frac{n}{m}$ or $\frac{n}{2}$ in this case) in the system. Obviously, $L_{over}(t) \geq L_{under}(t)$. Let $d_t = |L_1(t) - L_2(t)|$.
The BALANCE algorithm moves every job $j \in B_{over}(t)$ to the other machine with probability $\frac{d_t}{2L_{over}(t)}$. The idea is to achieve an equal expected load on both machines at the end of each step: $E[L_1(t+1)|L_1(t), L_2(t)] = \frac{n}{2}$. So, each job on *over* moves with probability $\frac{d_t}{2L_{over}(t)}$ and we get $L_{over}(t) \cdot \frac{dt}{2L_{over}(t)} = \frac{dt}{2}$ expected movements, as desired.
We would like to show that after expected $O(\log \log n)$ steps the system will reach an equilibrium. For the proof we use the following lemma.

Lemma 9.4.1 ***(Chernoff)*** *Let $z_1, ..., z_n$ be independent random binary variables and $Z = \sum_i z_i$, where $p = \sum_{i=1}^{n} \frac{E[z_i]}{n}$ and $\widehat{p} = \frac{1}{n}\sum_{i=1}^{n} z_i$. Then,*

$$(1) \quad P[p \leq \widehat{p} + \sqrt{\tfrac{2p\ln(1/\delta)}{n}}] \geq 1-\delta \qquad p \in [0,1]$$

$$(2) \quad P[p \geq \widehat{p} - \sqrt{\tfrac{3p\ln(1/\delta)}{n}}] \geq 1-\delta \qquad p \in [\tfrac{\ln(1/\delta)}{3n}, 1]$$

$$(3) \quad P[p \geq \widehat{p} - \tfrac{2\ln(1/\delta)}{n}] \geq 1-\delta \qquad p \in [0, \tfrac{\ln(1/\delta)}{3n}]$$

Theorem 9.6 *The BALANCE algorithm terminates within expected $O(\ln\ln n)$ steps.*

Proof. Let k be an upper bound on the number of steps of the BALANCE algorithm until it reaches Nash equilibrium. We divide our proof into two stages:

1. $dt > 3\ln(\frac{1}{\delta'})$
2. $dt \leq 3\ln(\frac{1}{\delta'})$

where $\delta' = \frac{\delta}{2k}$ and δ indicates the probability of the algorithm failing to end within k time steps.

First we show that the first stage terminates with probability $1 - \frac{\delta}{2}$ within $O(\ln\ln n)$ steps. By lemma 9.4.1 (2) for every step $t \leq k$, $d_{t+1} \leq \sqrt{3d_t\ln(\frac{1}{\delta'})}$ with probability $1 - \frac{\delta'}{2}$: represent each job on *over* as a binary random variable with $P(1) = \frac{d_t}{2L_{over}(t)}$. Thus, $p = \sum_{i=1}^{n} \frac{E[z_i]}{n} = \frac{d_t}{2L_{over}}$ and $\widehat{p} = \frac{1}{n}\sum_{i=1}^{n} z_i$ is the ratio of jobs that were actually moved in this step.

Since $d_1 \leq n$ we get

$$\begin{aligned}
d_{t+1} &\leq \sqrt{3d_t\ln(\tfrac{1}{\delta'})} \\
&= \sqrt{d_t}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&\leq \sqrt{\sqrt{3d_{t-1}\ln(\tfrac{1}{\delta'})}}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&= \sqrt{\sqrt{d_{t-1}}}\cdot\sqrt{\sqrt{3\ln(\tfrac{1}{\delta'})}}\cdot\sqrt{3\ln(\tfrac{1}{\delta'})} \\
&\vdots \\
&\leq n^{\frac{1}{2^{t+1}}}\cdot 3\ln(\tfrac{1}{\delta'})
\end{aligned}$$

For $t = O(\ln\ln n)$, $n^{\frac{1}{2^t}} = O(1)$ and therefore, $d_t \leq 3\ln(\frac{1}{\delta'})$ and the first stage terminates within $O(\ln\ln(n))$ steps. This situation remains with high probability until the k^{th} step.

In the second stage exactly $\frac{d_t}{2}$ jobs will be displaced in one step with probability $O(\frac{1}{\sqrt{\log(k)}})$, so the expected number of steps will be $O(\sqrt{\log k})$.
Summing the two stages we have $k = O(\ln \ln n + \sqrt{\log k}) = O(\ln \ln n)$. □

Unfortunately, the BALANCE algorithm does not assure that players will indeed act according to their best response at each step. For example, consider the setting in figure 9.1 at time t. What is the point of view of a single user on *over* (M_1)? Each user on M_1 will migrate to M_2 with probability $\frac{d_t}{2L_{over}(t)} = \frac{200}{800} = \frac{1}{4}$. Hence, for a single user on M_1:

- The expected load on M_1 without him is $(400-1) - \frac{1}{4}(400-1) = 300 - \frac{3}{4}$.
- The expected load on M_2 is $200 + \frac{1}{4}(400-1) = 300 - \frac{1}{4}$.

and the user prefers to remain on M_1. This provides an incentive for users to "cheat" and deviate from the joint strategy.
We now try to establish an algorithm which prevents such behavior.

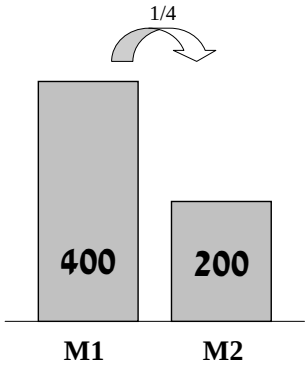


Figure 9.1:

The NashTwo Algorithm

Consider any stage before reaching Nash equilibrium where $L_1 = k+d$ and $L_2 = k-d$ i.e. $d_t = 2d$. We examine w.l.o.g a single user on L_1. We want to define the probability of migration p such that a single user will see an expected identical load on both machines (when excluding himself):
$(L_1 - 1)(1-p) = L_2 + (L_1 - 1)p$
$\Updownarrow$
$p = \frac{(L_1 - L_2) - 1}{2(L_1 - 1)} = \frac{d_t - 1}{2(L_1 - 1)}$

The NashTwo algorithm moves jobs from *over* to *under* with probability p.

Lemma 9.4.2 *At every stage Algorithm NashTwo is a Nash migration policy.*

Proof. Again we compare the loads on the two machines when excluding one user on the overloaded machine.
$L_1 = (k+d-1)(1-\frac{2d-1}{2(k+d)-2}) = n - \frac{1}{2}$
$L_2 = k - d + (k+d-1)(1-\frac{2d-1}{2(k+d)-2}) = n - \frac{1}{2}$
Therefore, each user on the overloaded machine cannot gain by deviating from the joint strategy. Users on the underloaded machine, like before, can obviously only lose by rerouting to the overloaded machine. □

It can be shown (in a different scribe...) that NashTwo converges at a similar rate as that of BALANCE.

9.4.2 Multiple Identical Machines

In the final subsection we extend the results to multiple identical machines. We seek a goal of dividing the load equally among the machines, i.e. to achieve a load of $\overline{L} = \frac{n}{m}$ which we assume to be an integer value. For each machine M_i, define $d_i(t) = L_i(t) - \overline{L}$, the difference between the current and the average load on machine M_i. Next, for each time t partition the machines into two disjoint sets,

- $Under(t) = \{M_i | d_i(t) < 0\}$
- $Over(t) = \{M_i | d_i(t) \geq 0\}$

We define $d_t = \sum_{i \in Over(t)} d_i(t)$.
Using these definitions the MultipleBALANCE algorithm proceeds as follows. Each user j on machine $M_i \in Over(t)$ determines whether it should move with probability $\frac{d_i(t)}{L_i(t)}$, and if so, selects its target underloaded machine $M_k \in Under(t)$ with probability $\frac{|d_k(t)|}{d_t}$.

Lemma 9.4.3 *(without proof) Let $Z_1, ..., Z_n$ be an i.i.d random binary variables with $P(Z_i = 1) = p$ and $Z = \sum_i Z_i$. Then, $P(Z = \lceil pn \rceil) \geq \frac{1}{e}\frac{1}{\sqrt{2\pi\lceil pn \rceil}}$. If $pn = q$ is an integer then, $P(Z = q) \geq \frac{1}{\sqrt{2\pi q}}$.*

Theorem 9.7 *The MultipleBALANCE algorithm convergence within expected $O(\log\log n + \log^2 m)$ steps.*

Proof. Let k be an upper bound on the number of steps of the MultipleBALANCE algorithm until it reaches Nash equilibrium. We divide our proof into two stages:
1. While there exists a machine M_i such that $d_i(t) > 3\ln(\frac{1}{\delta'})$
2. $\forall i$, $d_i(t) \leq 3\ln(\frac{1}{\delta'})$
where $\delta' = \frac{\delta}{3mk}$ and δ as before.
Similarly to the proof of the convergence time of the BALANCE algorithm, we get that for any machine M_i and time t, $d_i(t+1) \leq \sqrt{3d_k(t)\ln(\frac{1}{\delta'})}$. As before, using the fact that $d_i(1) \leq n$ and the mentioned computations will provide us the bound of $O(\ln\ln n)$ on the expected number of steps in the first stage.

During all of the second phase $\forall i$ it holds that $d_i(t) \leq 3\ln(\frac{1}{\delta'})$. Denote the number of unbalanced machines at time t as $Unbalanced(t)$. Note that $Unbalanced(t)$ can only decrease over time since once a machine is balanced it won't be changed.
We examine substages of the second phase. First we deal with the system when $Unbalanced(t) \geq \beta \log^{1.5}(\frac{1}{\delta'})$ with $\beta > 0$ a constant which will be specified from the proof. By lemma 9.4.3, since $d_i(t) \leq 3\ln(\frac{1}{\delta'})$, each unbalanced machine terminates in one step with probability $q = \Theta(\frac{1}{\sqrt{\ln(\frac{1}{\delta'})}})$. According to the standard properties of a binomial distribution, the expected number of machines in $Under(t)$ is at most $\frac{1}{2}Unbalanced(t-1)$. Since machines become balanced with probability $q < 0.01$, the expected number of machines in $Over(t)$ is $E[|Over(t)|] \geq 0.49Unbalanced(t-1)$. Let $O_i(t)$ be a variable indicating whether machine $M_i \in Over(t)$, thus we can write that $|Over(t)| = \sum_{i=1}^{m} O_i(t)$. The variables $O_i(t)$ are negatively associated and therefore we can apply the Chernoff bound on the number of overloaded machines as follows.

$$\begin{aligned} |Over(t)| &\geq 0.49Unbalanced(t-1) - \sqrt{3Unbalanced(t-1)\ln\tfrac{1}{\delta'}} \\ &\geq 0.48Unbalanced(t-1) \end{aligned}$$

for $Unbalanced(t-1) \geq \gamma(\log(\frac{1}{\delta'}))$, for some constant $\gamma > 0$ (with probability $1-\delta'$). The expected number of machines that become balanced at time t is at least $q|Over(t-1)|$. Note that each overloaded machine ($\in Over(t-1)$) becomes balanced with probability q independently and therefore we can apply the Chernoff bound on the number of new balanced machines as follows.

$$\begin{aligned} Unb(t) - Unb(t-1) &\geq 0.48Unb(t-1) - \sqrt{3qUnb(t-1)\log\tfrac{1}{\delta'}} \\ &\geq 0.47qUnb(t-1) \end{aligned}$$

for $Unbalanced(t-1) \geq \beta(\log^{1.5}(1/\delta'))$, for sufficiently large constant β (with probability $1-\delta'$). Consequently, after $O(\frac{\log m}{q}) = O(\log m\sqrt{\log(1/\delta')})$ the first substage terminates. Now, when $Unbalanced(t) < \beta\log^{1.5}(1/\delta')$, the number of unbalanced machines is only $O(\log^{1.5}(1/\delta'))$. Recall the second stage in the proof of the convergence time of the BALANCE algorithm. In a similar manner, it implies that after $O(\log^{1.5}(1/\delta')\cdot\sqrt{\log(1/\delta')}) = O(\log^2(1/\delta'))$ the remaining machines will be balanced. To conclude, we sum up our results to get,

$$\begin{aligned} k &= O(\log\log n + \log m\sqrt{\log(1/\delta')} + \log^2(1/\delta')) \\ &= O(\log\log n + \log m\sqrt{\log(mk/\delta)} + \log^2(mk/\delta)) \\ &= O(\log\log n + \log^2(m/\delta)) \end{aligned}$$

□

Note 9.8 *The component* $\log^2$ *can be improved to* $\log^{1.5}$*. Also, we can get a bound on the expected time of* $O(\log\log n + \log^{1.5} m)$ *(without dependency on* δ*).*

9.5 The End

Nothing lasts forever.

Computational Learning Theory **Spring Semester, 2003/4**

Lecture 10: Mechanism Design and Social Choice

Lecturer: Yishay Mansour *Scribe: Yair Halevi, Daniel Deutch, Ami Koren*

10.1 Mechanism Design

10.1.1 Motivation

So far we've seen various types of games, and investigated how they work. In this lecture we'll ask the opposite question: We want to reach a certain result. How should we plan a game that will lead to this result? This area is called **Mechanism Design**. In the second part of the lecture we will show that designing such a mechanism, even under simple assumptions, is sometimes impossible.

At any game, the rules (mechanism, institute) by which the game is taking place have a profound effect on both the way the players (agents) play, and on the decisions eventually taken. Examples:

- Auction – Are the bids given by sealed envelopes, or is it a public auction, with bids given orally? The method of offering the bids effects what participants learn on each other, the bids they give, and eventually the result of the auction.
- Scheduling – The scheduling-policy, effects which jobs are sent, and when.
- Public project (new highway, new library, park, defense system, etc.) – The way we spread the costs of the project across the society, effect the decision of whether the project is undertaken or not.

The aim is to plan a mechanism that guarantees certain objectives, based on the following assumptions:

- The agents have a strategic behavior. That is, they have a benefit function they want to maximize
- The agents have **private information** known only to them, that effect their decision.

For Example, Negotiation between a buyer and a seller.

Each side has a value of the deal (private information)

The seller will claim that his value is higher than it's real value (Increase the price)

The buyer will claim that his value is lower then it's real value (decrease the price)

Sample Questions:

- Plan a mechanism in which the negotiation takes place, that will cause an *efficient* merchandizing. That is, that a successful trade will take place whenever the buyer valuation exceeds the valuation of the seller.
- Is there an efficient mechanism where both the buyer and the seller agree to participate voluntarily?
- Elections – How do we plan fair elections. That is, how do we cause voters to vote by their real preferences, without lying?

Building blocks:

- Usually there is a central institution that governs the game
- Messages (For example, envelopes in auction) – These are the means of interaction with the agents. On literature, also called called types, strategies, or states.
- Payments – Optionally, we can tax or subsidize agents, in or to create incentives for a desired behavior.

Negative results:
We will show that in some cases, it is impossible to achieve all the following goals:

- Efficient result (maximum overall benefit)
- Voluntary participation
- Budget balancing (of tax/subside payments)

10.1.2 Formal Model

- Agents – $N = \{1..n\}$
- Decisions - $d \in D$
- Private information for agent i – $\theta_i \in \Theta_i$

- Utility function – $v_i : D \times \Theta_i \longrightarrow \mathbb{R}$ – The benefit of player i with private information θ_i from decision $d \in D$
- Preferences – $v_i(d, \theta_i) > v_i(\grave{d}, \theta_i)$ means that agent i prefer decision d over decision $\grave{d}$

Example 1 – Public project
The society want to decide whether or not to build a certain public project

- Cost of project – c (cost for participant - $\frac{c}{n}$)
- Decision – $D = \{0, 1\}$ (1–do / 0–don't do the project)
- Benefit of agent i from doing the project – θ_i.
- Benefit of agent i –

$$v_i(d, \theta_i) = d(\theta_i - \frac{c}{n}) = \begin{cases} 0 & \text{d=0} \\ \theta_i - \frac{c}{n} & \text{d=1} \end{cases}$$

Example 2 – Allocating a private product
An indivisible good is to be allocated to one member of the society. For instance, an enterprize is to be privatized. We have $N = \{1..n\}$ potential buyers. We want to give the product to one of them.

- Decision – $D = \{1..n\}$
- Benefit of agent i if achieving the product – θ_i
- Benefit of agent i –

$$v_i(d, \theta_i) = I(d = i) \cdot \theta_i$$

10.1.3 Decision rules and efficiency

The society would like to make a decision that maximizes the overall benefit of it's members. For example:

- For the public project, we will decide to build the project if the sum of benefits over all agents is more then it's cost
- For the private product, we might want to give it to the agent that have the maximal benefit from it

We will now phrase it formally.

From now on, we assume $\Theta = \Theta_1 \times \ldots \times \Theta_n$

Definition – Decision rule

$$d : \Theta \longrightarrow D$$

Definition – Efficiency

A decision d is efficient if it maximize the overall benefit:

$$\forall \theta \in \Theta, \grave{d} \in D : \sum_i v_i(d(\theta), \theta_i) \geq \sum_i v_i(\grave{d}, \theta_i)$$

For example:

- The public project should be done only if the overall benefit is higher then the cost: $d = 1 \Leftrightarrow \sum_i v_i(1, \theta_i)) \geq \sum_i v_i(0, \theta_i)) \Leftrightarrow \sum_i \theta_i - c \geq 0 \Leftrightarrow \sum_i \theta_i \geq c$
- Private product – The efficient decision is $d = maxarg_i\{\theta_i\}$

Payments

The definition of efficiency as the basis for decision making has a problem: We base it on a private information, which we don't actually have. We could ask the agents to give us their private value, but the answers will often be dishonest or deceptive.
For instance, in the public project, an agent with $\theta_i < \frac{c}{n}$ has an incentive to underreport his value, and claim he has no profit from the project, and hence try to manipulate the decision to his own benefit. In the same way, an agent with $\theta_i > \frac{c}{n}$ has an incentive to overreport his value. This could result in wrong decision. Other mechanisms of decision aimed to bypass this problem, such as voting and decide whether to build the project by the majority's vote, could also result in a decision which is not efficient.
The question is, can we create incentives for the agents to reveal their real private information? Many times the answer is yes. We can balance their interests by putting taxes (to reduce the profit) or by subsidizing agents (to reduce the loss). That is done by a **payment function**, which indicates how much each agent receives:

$$t_i : \Theta_i \longrightarrow \mathbb{R}$$

$$t : \Theta \longrightarrow \mathbb{R}^n$$

Definition – Social Choice Function
A social choice function is a pair of decision and payment:

$$f(\theta) = (d(\theta), t(\theta))$$

From now on, we will assume $\hat{\theta}$ to be the value declared by the agents, while θ is their true private value.

The utility that agent i receives is based on it's benefit from the decision taken, plus the payment he receives from the mechanism (both are based on it's declared value):

$$u_i(\hat{\theta}, \theta_i, d, t) = v_i(d(\hat{\theta}), \theta_i) + t_i(\hat{\theta})$$

A utility function at this form, where all arguments are concave except the last which is linear, is called Quasi-linear.

Definition – Balanced and Feasible payments

The payments are:

- Feasible, if $\forall\theta : \sum_i t(\theta) \leq 0$
- Balanced, if $\forall\theta : \sum_i t(\theta) = 0$

We want the payments to be balanced. If the payments are feasible and not balanced, we have a surplus we have to waste, or return to outside source. We can not return the change to the society, because the only way to do it is by changing the payment functions.

Balance is an important property if we wish (d, t) to be efficient rather then just d. If payments are feasible and not balanced, then there is some loss to the society relative to an efficient decision with no transfers.

Definition – Mechanism (M, g)

- Messages of agents: $(m_1, \ldots, m_n) \in M$, while $M = M_1 \times \ldots \times M_n$

- Outcome function: The outcome function $g : M \longrightarrow D \times \mathbb{R}^n$ is defined as $g(m) = (g_d(m), g_{t,1}(m), \ldots, g_{t,n}(m))$

  While:

  - $g_d : M \longrightarrow D$ is the decision function
  - $g_{t,i} : M \longrightarrow \mathbb{R}$ is the transfer function of agent i

  For example, First price auction:

- Benefit of agent i, if he achieves the product, is m_i
- Outcome:

$$g_d(m) = maxarg\{m_i\}$$

$$g_{t,i}(m) = \begin{cases} 0, & No\ win \\ -m_i, & Win \end{cases}$$

Mechanism Design and Dominant strategies

Definition – Dominant strategy
A strategy $m_i \in M_i$ is a *dominant strategy* at $\theta_i \in \Theta_i$ if it is superior to all other player strategies, regardless of other players strategies:

$$\forall m_i, m_i' \in M : v_i(g_d(m_{-i}, m_i), \theta_i) + g_{t,i}(m_{-i}, m_i) \geq v_i(g_d(m_{-i}, m_i'), \theta_i) + g_{t,i}(m_{-i}, m_i')$$

For example: Confess at the prisoner dilemma.

A dominant strategy is a compelling property of a mechanism, if it exists. It allows us a better prediction of which strategies will be employed by the agents. However, because it is such a strong property, it exists only at narrow space of problems.

Definition

A Social Choice Function $f = (d, t)$ is *implemented in dominant strategies* by a mechanism (M, g) if:

- There are functions $m_i : \Theta_i \longrightarrow M_i$ such that for any agent i with strategy $\theta_i \in \Theta$, $m_i(\theta_i)$ is a dominant strategy
- $\forall \theta \in \Theta : g(m(\theta)) = f(\theta)$

10.1.4 Direct Mechanism and Revelation theorem

We'll now show that if there is a dominant strategy, there is no need to use the complicated (M, g) mechanism. There is an equivalent and much simple method, called *Direct Mechanism (DM)*

Definition

A social choice function f can be viewed as a mechanism, where $M_i = \Theta_i$, and $g = f$. That is called *Direct Mechanism(DM)*

Definition

DM $f(d, t)$ is *dominant strategy incentive compatible (DS IC)* if for each agent i with strategy $\theta_i \in \Theta_i$, $\theta_i \in M_i$ is a dominant strategy at $\theta_i \in \Theta_i$
A social choice function f is *strategy proof* if it is dominant strategy incentive compatible.

In other words, saying that DM is dominant strategy incentive compatible means that telling the truth is the dominant strategy.

Theorem 10.1 ***Revelation principle for dominant strategies***
If a mechanism (M, g) implements a social choice function $f = (d, t)$ in dominant strategies, then the DM f is dominant strategy incentive compatible

Proof:
Since (M, g) implements f in dominant strategies, then there are $m(\theta) = (m_1(\theta), \ldots, m_n(\theta))$ such that:

$$\forall \theta : g(m(\theta)) = f(\theta)$$

And also:

$$\forall m_i, \hat{m}_i = m_i(\theta_i)$$

□

Negative Results:
We will see later that there is no voting method between 3 or more candidates that guarantees that the voters will tell their true preferences.

Theorem 10.2 – *Grove method*
If:

- *d is an efficient decision rule*

- *There is a function* $h_i : \Theta_{-i} \longrightarrow \mathbb{R}$ *such that:*
  $t_i(\theta) = h_i(\theta_{-i}) + \sum_{j \neq 1} v_j(d(\theta), \theta_j)$

then (d, t) *is dominant strategy incentive compatible*

Remark: Under some additional conditions, this is the only way to define dominant strategy incentive compatible.

Proof:
By way of contradiction, suppose d is an efficient decision rule, and that for each agent i there are h_i that define together $t : \Theta \longrightarrow \mathbb{R}^n$, But (d, t) is not dominant strategy incentive compatible.
Then, there is an agent i, and states θ and θ_i' such that θ_i' is more beneficial to agent i:

$$v_i(d(\theta_{-i}, \theta_i'), \theta_i) + t_i(\theta_{-i}, \theta_i') > v_i(d(\theta), \theta_i) + t_i(\theta)$$

We'll expand t_i explicitly, and write d' instead of $d(\theta_{-i}, \theta_i')$ for simplicity:

$$v_i(d', \theta_i) + h_i(\theta_{-i}) + \sum_{j \neq 1} v_j(d', \theta_j) > v_i(d(\theta), \theta_i) + h_i(\theta_{-i}) + \sum_{j \neq 1} v_j(d(\theta), \theta_j)$$

Therefore:

$$\sum_{j=1}^{n} v_j(d', \theta_j) > \sum_{j=1}^{n} v_j(d(\theta_{-i}, \theta_i), \theta_j)$$

And d' contradicts the efficiency of $d(\theta)$. The conclusion is that (d, y) is dominant strategy incentive compatible.

□

Clark Mechanism

Clark suggested a specific implementation of h_i

$$h_i(\theta_{-i}) = -max_{d \in D}\{\sum_{j \neq i} v_j(d, \theta_j)\}$$

which is actually the decision that would have been taken if agent i had not participated. Therefore, t_i is defined as follows:

$$t_i = \sum_{j \neq i} v_j(d, \theta_j) - max_{d \in D}\{\sum_{j \neq i} v_j(d, \theta_j)\}$$

Properties of Clack mechanism:

- The price is non-positive. That means that the agents pay to the mechanism, and assures *feasible payments*.
- If agent i does not effect the decision, then $t_i(\theta) = 0$
- $t_i(\theta)$ can be thought of as loss for the other agents

Example 1 – Second price auction
We give the products to the agent with the highest bid. How much should we charge for it?

- $d(\theta) = maxarg\{\theta_i\}$
- $v_i(d, \theta_i) = \begin{cases} 0 & d \neq i \\ \theta_i & d = i \end{cases}$
- For the agent with the highest value:
  - $\sum_{j \neq 1} v_j(d(\theta), \theta_j) = 0$
  - $h_i = -v_{2_{nd}}(d_{2_{nd}}, \theta_{2_{nd}}) = -\theta_{2_{nd}}$
  - $\Rightarrow t_i(\theta) = -\theta_{2_{nd}}$
- The participation of all other agents does not effect the decision, i.e. $t_i(\theta) = 0$

Explanation: The winner is the agent with the highest bid, and he pays the 2^{nd} highest offer. No other payments are done.

Example 2 – Building graph pathes
Given a graph $G = (V, E)$ with weights $\vec{c}$ on the each $e \in E$.

- Each $e \in E$ is an agent
- The cost of each agent is c_e (benefit is $-c_e$)
- The objective: to choose a minimal cost path
- $t_e(\vec{c}) = (Cost\ of\ shortest\ path - c_e) - (shortest\ path\ without\ e)$
  $t_e(\vec{c}) = (shortest\ path\ without\ e) - (cost\ of\ shortest\ path\ with\ e)$

Budget balancing vs. Incentive Compatible

It is not enough to require efficiency of decision. The society should also aspire to balanced payments. Transfer functions that are not balanced, cause waste, and it can be considerable. An example for it can be seen in the public project case:

Consider two agents. Project cost is $c = \frac{3}{2}$, and $\Theta_1 = \Theta_2 = \mathbb{R}$.
According to Clark, there are h_1 and h_2. We'll check what we can conclude from the feasibility of the payments.

1. Assume $\theta_1 = \theta_2 = 1$,
   $d(1,1) = 1$, and therefore:

$$0 \geq t_1(1,1) + t_2(1,1) = h_1(1) + 1 + h_2(1) + 1$$
$$\Rightarrow -2 \geq h_1(1) + h_2(1)$$

2. Assume $\theta_1 = \theta_2 = 0$,
   $d(0,0) = 0$, and therefore:

$$0 \geq t_1(0,0) + t_2(0,0) \geq h_1(0) + h_2(0)$$

From both inequalities we can conclude that there must be either $-1 \geq h_1(1) + h_2(0)$ or $-1 \geq h_1(0) + h_2(1)$. Since $d(1,0) = d(0,1) = 0$, in one of the cases $(0,1)$ or $(1,0)$ the payments are negative. That means that: a. to have a dominant strategy incentive compatible mechanism with efficient decision rule, one cannot satisfy balance. b. In some cases, there are payments without a project.

10.2 Social Choice

Social choice is the general problem of mapping a set of multiple individual preferences to a single social preference, that will best reflect the aggregate individual preferences. Common examples for such scenarios are public elections, voting, etc. While in mechanism design we have shown how to select a mechanism that will exhibit desired behavior, for social choice we shall show that given very simple and reasonable requirements, no such mechanism exists.

We shall show two theorems for two slightly different social choice scenarios: Arrow's Impossibility Theorem and the Gibbard-Satterthwaite Theorem.

10.2.1 Arrow's Impossibility Theorem

Arrow's Impossibility Theorem deals with social ordering. Given a set of alternatives $\mathcal{A} = \{A, B, C, \ldots\}$, a *transitive preference* is a ranking of the alternatives from top to bottom, with ties allowed. Given a set of individuals (a society, so to speak), a *social preference function* is a function associating any tuple of personal transitive preferences, one per individual, with a single transitive preference called the *social preference*.

Definition A **Transitive Preference** is an ordering of $\mathcal{A}$ with ties allowed.

For example: $A > B > C = D = E > F > G$.

Definition Given alternatives $\mathcal{A}$ and a set of individuals $\mathcal{N}$, a **Social Profile** is an association of a transitive preference per individual.

We will typically denote individual by numbers $1 \ldots N$, in which case a social profile will be represented by an N-tuple of transitive preferences. We will also represent a profile by a matrix, where each column is the transitive preference of single individual, ranked from top to bottom. For example:

1	2	3	4	5	6
A	A	B,C	C	A,B,D	A,B,C,D
C	B	-	A,B	-	-
B	D	A	-	-	-
D	C	D	D	C	-

Definition A **Social Preference Function** is a function associating each profile with a transitive preference, called the social preference.

1	2	3	4	5	6		Social
A	A	B,C	C	A,B,D	A,B,C,D		A
C	B	-	A,B	-	-	→	C
B	D	A	-	-	-		B,D
D	C	D	D	C	-		-

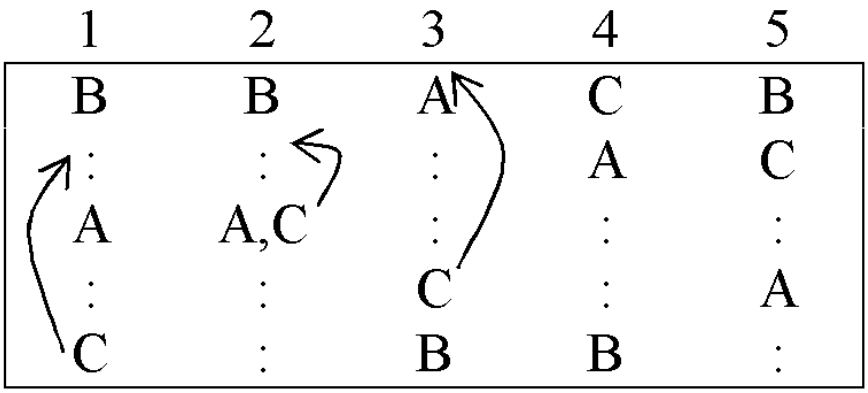


Figure 10.1: Moving C strictly above A without changing relative preference to B

Definition A social preference function respects **Unanimity** if the social preference strictly prefers α over β whenever all of the individuals strictly prefer α over β.

Definition A social preference function respects **I.I.A (independence of irrelevant alternatives)** if the relative social ranking (higher, lower or indifferent) of α and β depends only on their relative ranking in the profile.

Definition Given a social preference function, an individual n is a **Dictator** if the social preference strictly prefers α over β whenever n strictly prefers α over β. If a dictator exists, the social preference function is a **Dictatorship**.

Unless specifically stated, relationships (such as prefer, above, first, last, etc) are non-strict. We are now ready to present Arrow's Impossibility Theorem.

Theorem 10.3 ***Arrow's Impossibility Theorem*** *For 3 or more alternatives, any social preference function that respects transitivity, unanimity and I.I.A is a dictatorship.*

Proof: Assume a social preference function meeting the conditions set in the theorem. Let $B \in \mathcal{A}$ be chosen arbitrarily.

Claim 10.4 *For any profile where B is always either strictly first or strictly last in the ranking (for all individuals), the social preference must place B either strictly first or strictly last as well.*

Assume to the contrary that the social preference does not place B strictly first or strictly last. Then there exist two alternatives A and C (different from each other and B) such that in the social preference, $A \geq B$ and $B \geq C$. Since all individuals place B strictly first or strictly last, moving C strictly above A for an individual is possible without changing the relative preference between A and B or B and C for this individual. This is depicted in figure 10.1. Due to I.I.A, we conclude that moving C strictly above A for all individuals should not change the relative social preference between A and B, and between B and C, so we still have $A \geq B$ and $B \geq C$, which implies $A \geq C$ due to transitivity. But this contradicts unanimity, because now all individuals strictly prefer C over A.

Therefore, the social preference must place B strictly first or strictly last.

Claim 10.5 *There exists an individual n^* and a specific profile such that n^* can swing B from the strictly last position in the social preference to the strictly first position by changing his preference.*

We observe an arbitrary profile where B is strictly last for all individuals. Due to unanimity, B must be strictly last in the social preference. Now let the individuals from 1 to N move B from the strictly last position to the strictly first position successively. Due to the previous claim, in any stage B must be strictly first or strictly last in the social preference. Because it starts strictly last, and must end strictly first, there must be an individual whose change causes B to move from the former position to the latter. We denote this individual by n^*. Denote by profile I the profile just before n^* changes his preference, and by profile II the profile just after the change. Profile I is the profile mentioned in the claim, and n^* is the individual. This is depicted in figure 10.2.

Note that n^* will have this behavior for any profile where all individuals $i < n^*$ place B strictly first and all individuals $i \geq n^*$ place B strictly last. The reason is that the (strict) relative preferences between B and any other alternative in such a profile and in profile I are identical, so this must hold in the social preference, and thus B must still be strictly last in any such profile. The same is true for the changed profile and profile II, where B must be strictly first. Therefore the choice of n^* is only dependent on B, not the profile, and we can denote $n^* = n(B)$.

Claim 10.6 *n^* is a dictator for any pair of alternatives A and C that does not include B.*

Given any profile III where for n^*, $A > C$, create profile IV from profiles II and III by:

1. Start with profile II
2. Have n^* move A strictly above B, without changing the relative preferences among all alternatives other than A.

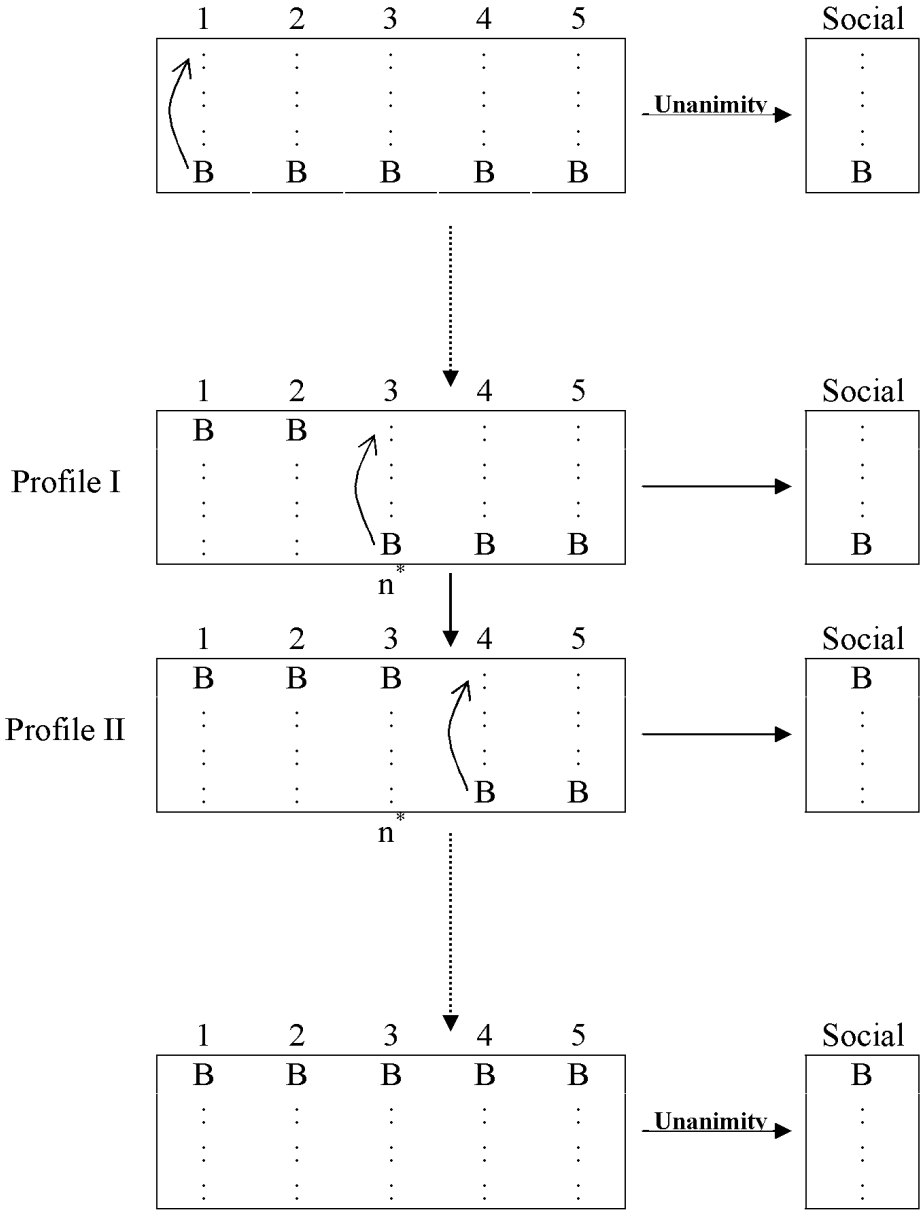


Figure 10.2: Existence of n^*

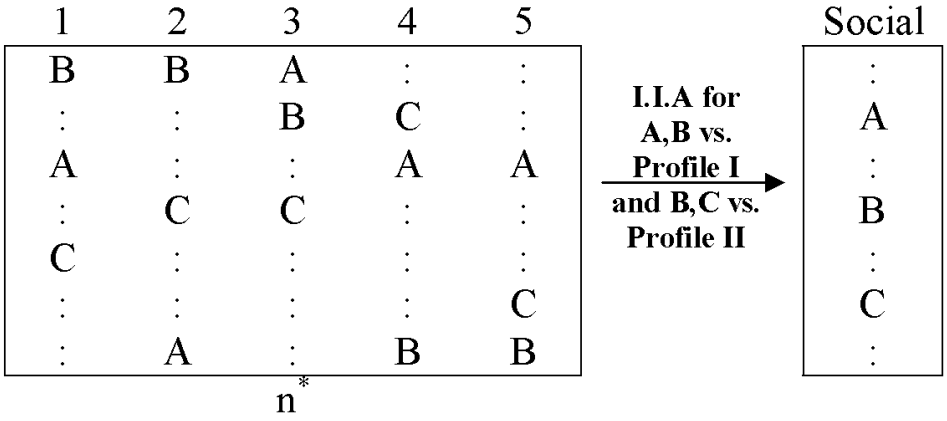


Figure 10.3: Profile IV

3. Have all other individuals change the preference between A and C to be identical to profile III, while B remains in it's profile II position.

In profile IV, the relative preference between A and B is identical to their relative preference in profile I for all individuals, and due to I.I.A we must have $A > B$ in the social preference for profile IV (because in profile I B is strictly last). The relative preference between C and B is in profile IV is identical to that of profile II for all individuals, thus (I.I.A) we must have $B > C$ in the social preference for profile IV (because in profile II B is strictly first). This is depicted in figure 10.3.

Therefore we must have $A > C$ in the social preference for profile IV. But the relative preferences between A and C in profiles III and IV are identical, so we must also have $A > C$ in the social preference for profile III. This is true for any profile III with n^* strictly preferring A over C, thus n^* is a dictator for A and C.

Claim 10.7 *n^* is a dictator for any pair of alternatives A and B.*

Choose a third alternative C. By the same construction above, there exists $n(C)$ that is a dictator for any pair of alternatives exclusive of C, such as A and B. But $n(B)$ definitely effects the relative social preference of A and B, so he is the only possible dictator for A and B, thus $n(C) = n(B)$. Therefore $n(B)$ is also a dictator for A and B.

We have shown that there is a single individual that is dictator for any pair of alternatives, thus the social preference function is a dictatorship. □

10.2.2 Gibbard-Satterthwaite Theorem

We shall now deal with an even simpler problem. The general scenario is similar to the one described previously. The difference is that we will only be interested in a single "most-

desired" alternative, instead of an entire social preference. Instead of looking for a social preference function, we are looking for an election mechanism.

Definition An **Election Mechanism** is a function mapping each social profile to a single alternative (the **elected alternative**) .

Definition An election mechanism M that decides an election is **Unanimous** if it elects alternative A whenever all individuals rate A as strictly first.

Definition A mechanism M that decides an election is defined to be a **strategy proof** one, when:

The dominant strategy of each voter is voting in the order of his real preferences ("telling the truth"). Namely, if the voter prefers candidate A over B, his dominant strategy will be to rank A above B. In other words, it is worthy for every voter to "tell the truth".

Definition A mechanism M that decides an election is defined to be a **dictatorial** one, when:

There exist a dictator, namely a voter v such that if v votes for candidate A, then A will win the election regardless of the other voters' votes.

Definition A profile is defined as a set that includes the preferences of all voters.

We are now ready to present the central theorem of this section.

Theorem 10.8 *(**Gibbard-Satterthwaite Theorem**) An election mechanism for 3 or more alternatives which is:*

- *Unanimous*
- *Strategy proof*

is a dictatorship.

This theorem will also be referred to as the **GS Theorem**. We precede the proof with a few lemmas.

Lemma 10.9 *Irrelevance Lemma*

Suppose that an alternative A is selected by the mechanism given a profile P. Then a modification of P which raises the ranking of an alternative X in the preference ranking of a single voter i, will cause either A or X to be selected by the mechanism.

Proof:

Suppose an alternative C, $C \neq X$, $C \neq A$ is elected. In P, A was elected, and the fact that i raised its vote for X caused C to be elected. There exist two cases:

1. If i prefers A to C, i would not raise his vote for X, even if the higher ranking of X is the truth for him, since then he causes C to be voted instead of A, in contradiction to strategy proof.

2. If i prefers C to A, i would never had reported his (maybe real) Low vote for X. He could gain profit from not reporting the truth, giving X a high vote, and thus C would be voted. Again, contradiction to strategy proof.

□

Lemma 10.10 *Unanimous last Lemma*

If an alternative B is ranked last by all voters, B is not elected.

Proof:

Assume B is elected. Then, for an alternative A, $A \neq B$, suppose that every voter, one at a time, raises his vote for A to the top of their priority. Then, by strategy proof, B is still elected, because otherwise for a voter interested in B to be elected, it would be profitable to him to vote for B last, thus not reporting the truth, and contradicting strategy proof.

But after all voters had raised their votes for A to the top, A must be elected, because of unanimity. Contradiction, thus the assumption that B was elected is not true, meaning B is not elected.

□

Proof of the GS Theorem:

Step 1

Begin with a profile P which is arbitrary in any sense, besides the ranking of B in it- every voter in P ranks B as his last vote. Thus, B is not elected, by "Unanimous last" Lemma. One voter at a time(arrange the voters in some order), have B "jump" from the bottom to the top of the voter's preferences. At the end of this process, all voters vote for B first, and thus B is elected(unanimity). So, let r be the first voter for which the change of his vote will cause B to be elected. r is called the **pivot for** B.

In all of the following tables, '?' stands for unknown, '...' means the sequence written before and after the dots (in the same line) is kept in between them.

Profile 1

1	2	.	.	r	.	.	n
B	.	.	B	K	?	?	?
?	?	.	.	.	.	.	?
?	?	.	.	.	.	.	?
?	.	.	?	B	B	.	B

$X \neq B$ is elected.

Profile 2

1	2	.	.	r	.	.	n
B	.	.	B	B	?	.	?
?	?	.	.	.	.	.	?
?	?	.	.	.	.	.	.
?	.	.	.	?	B	.	B

B is elected.
Consider profile 2:

1. **If any voter $i > r$ changes his vote, B is still elected**. Otherwise i, who does not want B to be elected, would change his vote to prevent it, in contradiction to strategy proof.

2. **If any voter $i \leq r$ keeps B ranked first, and changes the ranking of the other alternatives, B is still elected**. Otherwise i, who prefers B to be elected, would prefer to submit his original ranking even if his real preferences is represented by the modified one. Once again, contradiction to strategy proof.

Conclusion 1: **B is elected if the first r voters rank B first.**
Consider profile 1:

1. **If any voter $i < r$ changes his vote, B is still not elected**. Otherwise it is profitable for i to do this change, such that B will be elected. Contradiction to strategy proof.

2.**If any voter $i \geq r$ keeps B ranked last, and changes the ranking of the other alternatives, B is still not elected**.

Otherwise it is profitable for i to submit his original ranking(to prevent B from being elected) even if his real preferences are represented by the modified one. Contradiction to strategy proof.

Conclusion 2: **B is not elected if the voters r through n rank B last.**

We will show that the voter r is a dictator.

Step 2

profile 3

1	2	.	.	r	.	.	n
?	.	.	?	K	?	?	?
?	?	.	.	.	.	.	?
?	?	.	.	.	.	.	?
B	B	.	.	.	B	.	B

Raise K in profile 3 to the top position for all voters. K is now chosen, by unanimity. Now raise B to the top positions for voters 1... $r-1$:

profile 4

1	2	.	r-1	r	.	.	n
B	.	.	B	K	.	.	K
K	K	.	K	?	.	.	?
.	.	.	.	.	.	.	.
?	?	.	?	B	.	.	B

Since K was chosen in profile 3, and the modifications from profile 3 to 4 are only raises K or B, it is inferred from the Irrelevance Lemma above that either K or B are chosen in profile 4. But in profile 4 B is not chosen, by Conclusion 2 above (r through n ranked B last). Thus K **is chosen in profile 4**.

Now raise B to the second position for the voter r:

profile 5

1	2	.	r-1	r	r+1	.	n
B	.	.	B	K	K	.	K
K	K	.	K	B	?	.	?
?	.	.	.	.	.	.	?
?	?	.	?	?	B	.	B

K is still elected, since otherwise r would not have reported the change even if this is a change in his real preferences, in contradiction to strategy proof.

Reconsider profile 3.

Lemma 10.11 *In profile 3, K is elected.*

Proof:

Start with profile 3, and assume $G \neq K$ is elected. Raise B to the top position for the first $r-1$ voters. By Conclusion 2, B is not elected, and by the Irrelevance Lemma, G is still elected.

Now raise B to the second position in the voter r's ranking.

profile 6

1	2	.	r-1	r	r+1	.	n
B	.	.	B	K	?	.	?
?	?	.	?	B	?	.	?
?	.	.	.	.	.	.	?
?	?	.	?	?	B	.	B

B is elected in profile 6.

Assume B is not elected in profile 6. By the Irrelevance Lemma, G is still elected. By Conclusion 1, if we raise alternative B in the vote of r one step up, to the top position, then B is elected. In profile 6, r prefers B over $G \neq B$, so he would profit from falsely reporting B above K, in contradiction to strategy proof.

Now, in profile 6, raise K to the second place in the votes of voters 1 through $r-1$, and to the top position for voters $r+1$ through n.

Profile 7

1	2	.	r-1	r	r+1	.	n
B	.	.	B	K	K	.	K
K	.	.	K	B	?	.	?
?	.	.	.	.	.	.	?
?	?	.	?	?	B	.	B

B is still elected in profile 7:

Assume B is not elected.

Voters 1 through $r-1$ who want B to be elected will profit by not reporting the change(even if it is truthful). Voters $r+1$ through n who want B not to be elected will profit by falsely reporting the change. Thus B is elected in profile 7. But profile 7 = profile 5, and we proved above that $K \neq B$, is elected in profile 5, in contradiction. Thus, the assumption that $G \neq K$ is elected in profile 3 is not true. Meaning, K **is elected in profile 3**.

□

Step 3

Lemma 10.12 *Consider an arbitrary profile P where r ranks some alternative $K \neq B$ on top. Then K or B is elected.*

Proof:

First, modify the profile by moving B to the bottom for all voters. We get profile 3, and we showed that K is elected. Now, one voter at a time, restore the profile by raising B to its original position. By the Irrelevance Lemma, **either K or B is elected**.

□

Lemma 10.13 *Consider an arbitrary profile P where r ranks some alternative $K \neq B$ on top. Then K is elected.*

Proof:

Consider:

Profile 8

1	2	.	r-1	r	r+1	.	n
B	.	.	B	B	A	.	A
?	?	.	?	?	?	.	?
?	?	.	?	?	?	.	?
C	C	.	.	.	.	.	C

where $C \neq B$ and $C \neq K$. Again, similarly to step 1, have C jump in the ranking of the voters one by one, starting from voter 1 until a pivot m, the first for whom the election will become C is found. Symmetrically to step 2, the top choice of m is selected in profile 8. But from Conclusion 1, we know that B is chosen in profile 8. Meaning, the top choice of m in

profile 8 is B, meaning $m \leq r$. But a symmetric argument, starting with m and then finding r (Everything is done exactly the same, replacing m with r and B with C), will lead to the conclusion that $r \leq m$, and so $m = r$. So r, the pivot in respect to B, is also the pivot in respect to C. Using Lemma 10.10 for C instead of B, we obtain that K or C are elected in P. Thus, In P:

K or C are elected.

K or B are elected.

And we obtain:

K is elected in P.

□

For $K = B$, similar arguments show that r is a pivot for A, as well as C, and that B is elected. Hence, for each K, $K \neq B$ or $K = B$, if r ranks some alternative K on top, Then K is elected. Hence, r is a dictator.

□

Computational Game Theory **Spring Semester, 2003/4**

Lecture 11:June 8

Lecturer:Yishay Mansour *Scribe:Nir Yosef, Itamar Nabriski, Nataly Sharkov*

11.1 Introduction

In this lecture we consider **Combinatorial Auctions** (abbreviated *CA*), that is, auctions where instead of competing for a single resource we have multiple resources. The resources assignments and bids are defined on subsets of resources and each player has a valuation defined on subsets of the resource set he was assigned. The interesting cases here is when the valuation of a given set of resources is different form the sum of valuations of each resource separately (the whole is *different* from the sum of its parts). That could happen when we have a set of complementary products that is, each product alone is useless but the group has a significantly larger value (for example - left and right shoes). On the other hand we might have a set of substitutional products where the opposite takes place (for example - tickets for a movie - no use of having two tickets if you are going alone).
In these cases there is an importance for pricing groups of resources rather than single resources separately, i.e. in the absence of complementarity and substitutability (if every participant values a set of goods at the sum of the values of its elements), one should organize the multiple auction as a set of independent simple auctions, but, in the presence of these two attributes, organizing the multiple auction as a set or even a sequence of simple auctions will lead to less than optimal results, in such a case we use **Combinatorial Auctions**.

11.2 Preliminaries

Throughout this lecture, we shall consider single-side combinatorial auctions, that is, auctions with single seller and multiple buyers.
Any such auction must specify three elements:

- The bidding rules (i.e., what one is allowed to bid for and when).
- The market clearing rules (i.e., when is it decided who bought what and who pays what)
- The information disclosure rules (i.e., what information about the bid state is disclosed to whom and when).

We consider only one-stage, sealed-bid *CAs*: each bidder submits zero or more bids, the auction clears, and the results are announced.
The third element of the specification is thus straightforward: no information is released about other bidders' bids prior to the close of the auction. The first element of the specification is almost as straightforward: each bidder may submit one or more bids, each of which mentions a subset of the goods and a price. One has to be precise, however, about the semantics of the collection of bids submitted by a single bidder, because, as was mentioned, the bid for a group doesn't necessarily equal to the sum of bids of its elements.
Only the second element of the above specification, the clearing policy, provides choices for the designer of the *CA*. There are two choices to be made here: which goods does every bidder receive, and how much does every bidder pay? We address these below.

11.2.1 The model

- $N = \{1..n\}$ set of players.
- $S = \{1..m\}$ set of resources (products).
- Θ - set of players private information, player i has information $\theta_i \in \Theta_i$ which is the inner state he is currently in.
- D - Mechanisms decision space - each vector specifies resources allocation amongst the players. $D = \{< s_1..s_n > \mid (\forall i \neq j \ s_i \cap s_j = \emptyset) \wedge (\bigcup_{1 \leq i \leq n} s_i \subseteq S) \}$.
- $V = \{V_1..V_n\}$ - set of preference functions $V_i : D \times \Theta_i \rightarrow R$
  which is the value which player i attributes to every subset of S given its internal state θ_i.
- $\vec{t} = \{t_1..t_n\}$ - set of payments defined for each player by the mechanism
  $t : \Theta \rightarrow R^n$, $t_i(\Theta) \in R$.

Remark 11.1 ***Monotonicity*** *for every $s_1, s_2 \in S$ such that $s_1 \subseteq s_2$, the value attributed to s_2 will not be smaller to that of s_1. i.e. $s_1 \subseteq s_2 \Rightarrow V_i(s_1) \leq V_i(s_2)$ for any player i.*

11.2.2 Goals and assumptions

- Our goal will be guaranteeing **Efficiency** - find a *pareto-optimal* allocation, that is, no further trade among the buyers can improve the situation of some trader without hurting any of them. This is typically achieved by using an assignment which brings the sum of benefits to a maximum.
- An alternative goal - maximizing Seller's revenue (will not be discussed on this lecture)

- Assumption - **no-externalities** : Players' preferences are over subsets of S and do not include full specification of preferences about the outcomes of the auction (the resulting allocation). Thus, a player cannot express externalities, for example, that he would prefer, if he does not get a specific resource, this resource to be allocated to player X and not to player Y.

11.3 Mechanism Design for *CA*

In order to get an efficient allocation where for each player *telling the truth* is a dominant strategy we'll use the *VCG* mechanism.

11.3.1 *VCG* mechanism - definition

- Decision rule(resource allocation): $d =< s_1..s_n >\in D$ such that
  d $= ArgMax_{\,d\in D} \sum_i V_i(s_i, \theta_i)$. That is, the chosen allocation maximizes the sum of the declared valuations of the players.

- Payment scheme: $t_i(\theta) = \sum_{j\neq i} V_j(s_j, \theta_j) - VMax_{<s'_1..s'_n>|s'_i=\emptyset} \sum_{j\neq i} V_j(s_j, \theta_j)$. That is, each player receives a monetary amount that equals the sum of the declared valuations of all other players, and pays the auctioneer the sum of such valuations that would have been obtained if he had not participated in the auction.

Remark 11.2 *Note that A bidder knows his own inner state, but this information is private and neither the auctioneer nor the other players have access to it, thus both of the above are functions of the players' declarations rather than its inner state.*

In the previous lecture we've seen that this mechanism brings the social benefit (sum of all benefits calculated according to players' declarations) to a maximum while keeping truth-telling as a dominant strategy.

11.3.2 Problems with the implementation of *VCG* mechanism

The main problem we confront trying to implement *VCG* mechanism is a computational problem, as it turns out, finding such a maximum benefit allocation is a *NP-hard* optimization problem (moreover, in our case we need to calculate a maximum benefit allocation $n+1$ times) that, in the worst case, cannot be even approximated in a feasible way.
An additional problem is describing players' preferences: the domain of the preference function is the product of all subsets of S with player's internal state and as such, for a given state, its size is exponential in m.
Comment: the size of the domain mentioned above is under the assumption of no-externalities.

Without that assumption, the domain would have been much larger ($|D| \times |\Theta|$)

In the following sections we consider a simplified model of *CA* called *SMB* (single minded bidder) defined as:
For every player i there exists a single set $s_i \subseteq S$ which he wants and for which he is willing to pay the (non-negative) price c_i.

$$V_i(s) = \begin{cases} c_i & s_i \subseteq S \\ 0 & \text{otherwise} \end{cases}$$

We have a compact description for the players' preferences $< s_i, c_i >$, thus overcoming the second problem, next we'll see that even for that simplified model, implementing *VCG* i.e. finding maximal allocations, is NP-hard.

11.3.3 Reduction from IS

Claim 11.3 *Finding an optimal allocation on* CA *with* SMB *model is NP-hard*

Proof: We prove the claim by showing a reduction from the graph-theory problem of maximum independent set to a maximum allocation problem on *SMB* model: Given an undirected graph $G = (V, E)$ let us build an instance of *CA* as follows:

- $S = E$: every edge is considered as a resource
- $N = V$: every vertex is considered as a player
- for each player (vertex) i, define s_i as the set of all edges (resources) coming out of that vertex and $c_i = 1$.

For example, see following figure:

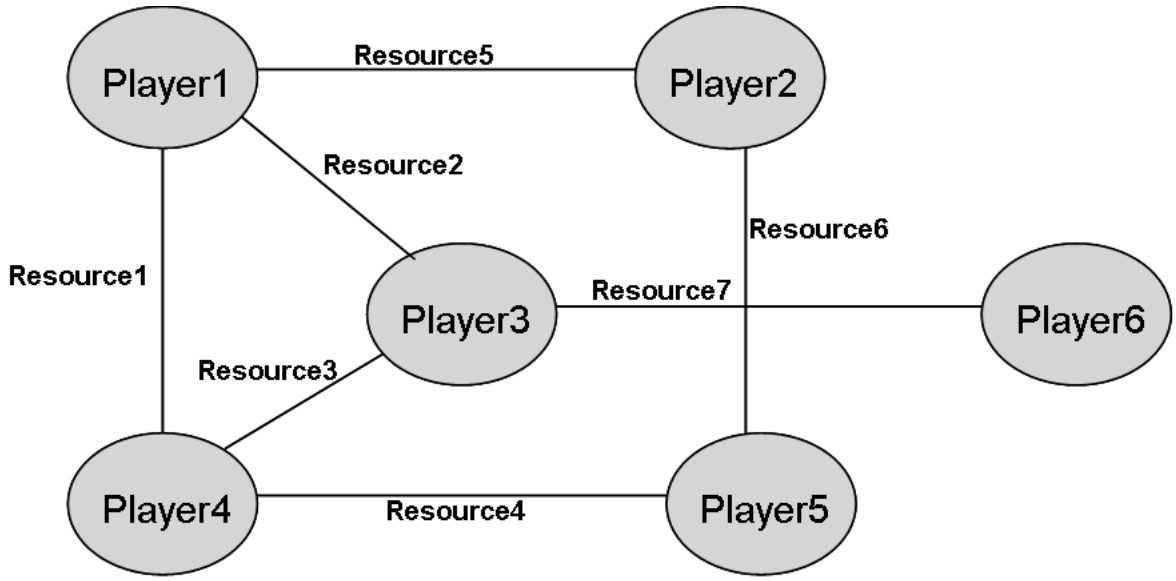


Fig.1 Reduction from IS on an undirected graph to finding optimal allocation on CA with SMB
For example: Player1 desired set of resources (s_1) is $\{2, 5, 1\}$

>From the definition of D above, it is easy to see that:

- any legal allocation defines an independent set (the set of all players(vertices) with a non-zero benefit) with the same value

- on the other hand, any independent set Δ defines a legal allocation (Allocate s_i for every player(vertex) i such that $i \in \Delta$) with the same value as well.

Thus, finding a maximal social benefit is equivalent to finding a maximum independent set. From the above reduction and since *IS* is in *NPC*, we conclude the same on the problem of finding an optimal allocation. □

Corollary 11.4 *Since we have $|E| \leq |V|^2$ resources and since no approximation scheme for IS has an approximation ratio of $|V|^{1-\epsilon}$ we get a bound of $\sqrt{m}$ on the approximation ratio for our problem where m is the number of resources.*

11.4 The greedy allocation

As we have seen, for all practical purposes, there does not exist a polynomial-time algorithm for computing an optimal allocation, or even for computing an allocation that is guaranteed to

be off from optimal by at most a constant, any given constant. One approach to meeting this difficulty is to replace the exact optimization by an approximated one. Next, we shall propose a family of algorithms that provide such an approximation. Each of those algorithms runs in a polynomial time in n, the number of single-minded bidders. Finally, we (unfortunately) see that the properties guaranteed by the mechanism (such as truthful bidding, to be defined later), disappear when using these approximated allocations.
(**comment** - *traditional analysis of established CA mechanisms relies strongly on the fact that the goods are allocated in an optimal manner*).
General description of the algorithms:

- First phase: the players are sorted by some criteria. The algorithms of the family are distinguished by the different criteria they use.

- Second phase: a greedy algorithm generates an allocation. Let L be the list of sorted players obtained in the first phase. The bid of the first player i_1 of L ($< s_{i_1}, c_{i_1} >$); is granted, that is, the set s_{i_1} will be allocated to player i_1. Then, the algorithm examines all other player of L, in order, and grants its bid if it does not conflict with any of the bids previously granted. If it does, it denies (i.e., does not grant) the bid.

11.4.1 First sort criteria: c_i

Claim 11.5 *Using a greedy algorithm, G_1, with c_i as a sort criteria would yield an approximation factor of m*

Proof:
$\Rightarrow$ The ratio is at least m, proof by example:
Suppose we have a set $N = \{1..n\}$ of players (*SMB*'s) and a set $S = \{1..m\}$ of resources where $m = n$. and suppose:

- Player 1 asks for all the resources and his value is $1 + \epsilon$, $[s_1 = S, c_1 = 1 + \epsilon]$

- $\forall 2 \leq i \leq n$ player i asks for resource i and his value is 1 , $[s_i = \{i\}, c_i = 1]$

In this case it follows that $OPT = m$ but $G_1 = 1 + \epsilon$

$\Leftarrow$ The ratio can be at most m because the value of the first player in a greedy allocation is higher than that of any player in OPT (follows immediately from the feasibility of OPT) $\square$

11.4.2 Second sort criteria: $\frac{c_i}{|s_i|}$

Claim 11.6 *Using a greedy algorithm, G_2, with $\frac{c_i}{|s_i|}$ as a sort criteria would yield an approximation factor of m.*

Proof:
$\Rightarrow$ The ratio is at least m, proof by example:
Assuming we have a set of two players and a set of resources similar to the above, suppose:

- Player 1 asks for resource 1 and his value is 1 $[s_1 = 1, c_1 = 1]$
- Player 2 asks for all the resources and his value is $m - \epsilon$ $[s_2 = S, c_2 = m - \epsilon]$

In this case it follows that $OPT = m - \epsilon$ but $Greedy = 1$

$\Leftarrow$ The ratio can be at most m:
>From the greediness property of G_2, for any subset s_i(requested by player i) that was allocated by OPT and not allocated by Greedy there exists at least one other subset which was previously allocated by G_2 and because of which s_i was not allocated.
Let us consider the following function defined on the subsets allocated by OPT:

$$\forall_{i \in OPT} \quad J(i) = \begin{cases} j: & (j \in G_2) \wedge (s_i \cup s_j \neq \emptyset) \quad i \notin G_2 \\ i & \text{otherwise} \end{cases}$$

Explanation: for any subset s_i (requested by player i) that was allocated by OPT and not allocated by G_2, we take $s_{J(i)}$ as a subset because of which s_i was not allocated. And, for any subset s_i which was allocated both by OPT and G_2 we take $J(i)$ to be equal to i

Now, from the above definition of J and from the feasibility and greediness of G_2, we can conclude ($\forall_{i \in OPT}$):

1. $s_i \cap s_{J(i)} \neq \emptyset$
2. $\frac{c_i}{|s_i|} \leq \frac{c_{J(i)}}{|s_{J(i)}|}$

>From which follows: $c_i \leq \frac{|s_i|}{|s_{J(i)}|} c_{J(i)} \leq |s_i| c_{J(i)}$

And finally:

$OPT = \sum_{i \in OPT} c_i \leq \sum_{i \in OPT} |s_i| c_{J(i)} \leq m \sum_{i \in OPT} c_{J(i)} \leq \sum_{j \in G_2} c_j = m \cdot G_2$

- The third inequality is due to the fact that OPT is feasible i.e.,
$(s_1, s_2 \in OPT) \rightarrow (s_1 \cap s_2 = \emptyset)$ $\square$

***Remark on notation**: for a player i and an algorithm ALG we say that $i \in ALG$ if the request of player i was granted by ALG*

11.4.3 Third sort criteria: $\frac{c_i}{\sqrt{|s_i|}}$

Claim 11.7 *Using a greedy algorithm, G_3, with $r_i = \frac{c_i}{\sqrt{|s_i|}}$ as a sort criteria would yield an approximation factor of $\sqrt{m}$*

Proof:
Consider the following two inequalities:

$G_3 = \sum_{j \in G_3} c_j \geq \sqrt{\sum_{j \in G_3} c_j^2} = \sqrt{\sum_{j \in G_3} r_j^2 |s_j|}$

- Because $\forall_{1<j<n}$, $c_j > 0$

$OPT = \sum_{i \in OPT} r_j \sqrt{|s_j|} \leq \sqrt{\sum_{i \in OPT} r_i^2} \sqrt{\sum_{i \in OPT} |s_i|} \leq \sqrt{m} \sqrt{\sum_{i \in OPT} r_i^2}$.

- The last inequality follows from: $(\forall i_1, i_2 \in OPT i_1 \neq i_2) \rightarrow (s_{i_1} \cup s_{i_2} = \emptyset)$

Thus it is enough to compare $\sqrt{\sum_{j \in G_3} r_j^2 |s_j|}$ and $\sqrt{\sum_{i \in OPT} r_i^2}$

Let us consider the function $J(i)$ as in the last proof. In the same manner we can conclude $\forall i \in OPT$:

1. $s_i \cap s_{J(i)} \neq \emptyset$
2. $r_i \leq r_{J(i)}$

>From the feasibility of OPT it follows that for every subset s_j allocated by G_3, there exists at most $|s_j|$ subsets which are allocated by OPT and rejected by G_3 because of s_j. Summing for all $i \in OPT$, we get:

$\sqrt{\sum_{i \in OPT} r_i^2} \leq \sqrt{\sum_{i \in OPT} r_{J(i)}^2} \leq \sqrt{\sum_{j \in G_3} r_j^2 |s_j|}$

And finally, we get:
$OPT \leq \sqrt{m} \sqrt{\sum_{i \in OPT} r_i^2} \leq \sqrt{\sum_{j \in G_3} r_j^2 |s_j|} \leq \sqrt{m} G_3$ □

11.5 Truthful Mechanism with Greedy Allocation in *SMB*

11.5.1 Greedy Allocation Scheme and *VCG* do not make a Truthful Mechanism in *SMB*

The following example illustrates a case where using G_2 and VCG doesn't yield a truthful mechanism (and simiraly for any G_i):

Player	$< s_i, v_i >$	$\frac{v_i}{\lvert s_i \rvert}$	t_i
R	$(\{a\}, 10)$	10	$8 - 19 = -11$
G	$(\{a, b\}, 19)$	9.5	0
B	$(\{b\}, 8)$	8	$10 - 10 = 0$

Since the t_i's represent the value gained by the other players in the auction minus the value gained by the other players had i not participated in the auction, R ends up with a lose of 11. Had R not been truthful and bid below 9.5 (G_2's $\frac{v_i}{|s_i|}$), he would be better off gaining 0. Thus in this case being truthful is not a dominant strategy for R and thus this mechanism is not truthful.
We now explore the conditions necessary for a truthful greedy allocation mechanism in *SMB*.

11.5.2 Sufficient Conditions for a Thruthful Mechanism in SMB

Let $\{g_1, ..., g_n\}$ denote the set of allocations the mechanism grants to each player. For brevity all bids and valuations are not labeled by the player index and all pretain to player i
Definition Exactness: Either $g_i = s$ or $g_i = \emptyset$.

In other words player i is allocated all the goods he bid for or none at all. There are no partial allocations.

Definition Monotonicity: $s \subseteq g_i$, $s^{'} \subseteq s$, $v^{'} \geq v \Rightarrow s^{'} \subseteq g_i^{'}$.

This means that if player i's bid was granted for bidding $< s, v >$ then his bid would also be granted for bidding $< s^{'}, v^{'} >$ where $s^{'} \subseteq s$ and $v^{'} \geq v$. Thus if a bid for a set of goods is granted then a bid (with at least the same amount of money) for a subset of the goods will be granted as well.

Lemma 11.8 *In a mechanism that satisfies* Exactness *and* Monotonicity*, given a bidder i, a set s_i of goods and declarations for all other bidders in the game, there exists a* critical value v_c *such that:*

$$v_i < v_c \Rightarrow g_i = \emptyset$$

$$v_i > v_c \Rightarrow g_i = s_i$$

Note that we do not know if i's bid is granted when $v_i = v_c$ and that v_c can be infinite and thus for every v, $g_i = \emptyset$.

Proof: Assume by contradiction our mechanism supports *Exactness* and *Monotonicity*, but a v_c as described above does not exist then either:

1. For a bid v_i by player i, $g_i \neq s$ and $g_i \neq \emptyset$. But this contradicts *Exactness*. Contradiction.

2. For two different possible bids of player i, v_1, v_2: $v_1 < v_2$ and $g_{i_1} = s$, $g_{i_2} = \emptyset$. But this contradicts *Monotonicity*. Contradiction.

□

Definition Critical: $s \subseteq g_i \Rightarrow t_i = v_c$

This has two meanings:

1. The payment for a bid granted to player i does not depend on his bid but on the bids of the other players.

2. The payment equals exactly to the (critical) value below which the bid will not be granted.

Definition Participation: $s \not\subseteq g_i \Rightarrow t_i = 0$

This implies that if you are not granted the goods you bid for, you will not incur any payments.

Lemma 11.9 *In a mechanism that satisfies* Exactness *and* Participation*, a bidder whose bid is denied has a profit of zero.*

Proof:
By *Exactness*, the bidder gets nothing and thus his income is zero. By participation his payment (expenditure) is zero. Thus $profit = income - expenditure = 0 - 0 = 0$.

□

Lemma 11.10 *In a mechanism that satisfies* Exactness, Monotonicity, Participation *and* Critical *a truthful bidder's profit is nonnegative.*

Proof:

If player i's bid is denied, we conclude be lemma 11.9 that i's profit is zero. Assume i's bid is granted and his type is $< s, v >$. Being truthful, i declaration is $d_i =< s, v >$. Thus i is allocated s and his income is v. By lemma 11.8 , since i's bid is granted, $v \geq v_c$. By *Critical*, i's payment is v_c, thus his profit is $v - v_c \geq 0$.

□

Lemma 11.11 *In a mechanism that satisfies* Exactness, Monotonicity, Participation *and* Critical, *a bidder i of type $< s, v >$ is never better off declaring $< s, v' >$ for some $v' \neq v$ than being truthful.*

Proof:

For player i, compare the case i bids truthfully $< s, v >$ and the case he bids $< v', s >$. Let g_i be the goods he recieves for $< s, v >$ and g_i' be the goods he recieves for $< s, v' >$. There are three cases:

1. If i's bid is denied for $< s, v' >$ (thus $g' \neq s$), then by lemma 11.9, his profit is zero and by lemma 11.10 his profit for $< s, v >$ is nonnegative and the claim holds.

2. Assume i's bid is granted both for $< s, v' >$ and $< s, v >$ thus $g_i' = s$, $g_i = s$. If both bids are granted then in both cases the player gets goods that he values to be worth v. In both cases the player pays the same payment v_c (by *Critical*).Thus profit is identical in both cases and the player is not better off lying.

3. Assume i's bid is granted for $< s, v' >$ but denied for $< s, v >$ thus $g_i' = s$, $g_i = \emptyset$. It must be that $v \geq v_c \geq v'$. By lemma 11.9, being truthful gives i zero profit. Lying gives him profit $v - v_c \leq 0$.

□

Lemma 11.12 *In a mechanism that satisfies* Exactness, Monotonicity *and* Critical, *a bidder i declaring $< s, v >$ whose bid is granted ($g_i = s$), pays a price t_i where $t_i \geq t_i'$ and t_i' being the price payed for declaring $< s', v >$ where $s' \subseteq s$.*

Proof:

Since $< s, v >$ was granted, by *Monotonicity*, so would $< s', v >$. By *Critical*, the price t_i' payed for $< s', v >$ satisfies: for any $x < t_i'$ the bid $< s', x >$ would be denied. By *Critical*, for any $x > t_i$ the bid would be granted. Thus, it must that $t_i' \leq t_i$.

□

Using the above lemmas we will prove the following central Theorem:

Theorem 11.13 *If a mechanism satisfies* Exactness*,* Monotonicity*,* Participation *and* Critical*, then it is a truthful mechanism.*

Proof:
Suppose player i's type is $< s, v >$, we prove he is not better off declaring $< s', v' >$:
By lemma 11.10 the only case we must consider is when declaring $< s', v' >$ yields positive profit to i and by lemma 11.9 this means that this bid was granted. Assume, therfore that $g'_i = s'$.

1. Assume $s \not\subset s'$. By *SMB* defintion, player i's income is zero (he got the bundle he doesn't want...). Since, by *Critical*, his payment is non-negative, his profit cannot be positive.

2. Assume $s \subset s'$. Being an *SMB* , i's income from s' is the same as from s. By lemma 11.12 it is evident that instead of declaring $< s', v' >$, i would not be worse off declaring $< s', v >$. By lemma 11.11 it is evident that $< s', v >$ is not better off than being truthful, or in other words declaring $< s, v >$.

□

11.5.3 A Truthful Mechanism with Greedy Allocation

We shall now describe a *payment scheme* that used with greedy algorithms of type G_i creates a truthful mechanism for SMB.
The mechanism proposed is for G_2, i.e. sorting bids by $\frac{v_i}{|s_i|}$. This can easily be adapted to G_1,G_3 or any sort of G_3 with a different norm with no added complexity.
The payment computation is done in parallel with the execution of G_i. Each payment calculation takes $O(n)$ and thus computing all the payments is $O(n^2)$. Since G_i takes $O(n \log n)$ the total running time is $O(n^2)$.

Definitions

- $AverageCost_i = \frac{v_i}{|s_i|}$
- $NextBidder(i) : N \longrightarrow N$, returns the first bidder following i (in the the sorted descending list of bids, that is $AverageCost_i \geq AverageCost_{NextBidder(i)}$) whose bid was denied, but would be granted had we removed i from the game. Defined Formally:

  $$NextBidder(i) = min\{i | i < i, s(i) \cap s(i) \neq \emptyset, \forall l, l < i, l \neq i, l \text{ granted} \Rightarrow s(l) \cap s(i) = \emptyset\}$$

- *Greedy Payment Scheme* (GPS). Let L be the sorted list created by G_i:

1. If $g_i = s_i$, i pays $AverageCost_{NextBidder(i)} \times |s_i|$ (if there is no next bidder payment is 0), else:
2. i pays 0.

Proposed Mechanism

Theorem 11.14 *G_i together with GPS comprise a truthful mechanism for the SMB.*

Proof:
We shall prove that G_i together with GPS satisfies *Exactness*, *Motonicity* , *Participation* and *Critical* and use Theorem 11.13 to conclude it is a truthful mechanism:

1. Exactness:
   By definiton of G_i.
2. Motonicity:
   For any G_i and a player i with bids of $< s, v >, < s', v' >$, if $g_i = s$, $s' \subseteq s$ and $v' \geq v$ then bidding $< s', v' >$ would put i in an equal or better place in L and thus $g'_i = s'$ as well.
3. Participation:
   By definiton of GPS.
4. Critical:
   For G_2 with GPS, but similarly for any type of G_i with a similar GPS, if player i bids $< s, v >$ and $g_i = s$ then i pays $AverageCost_{NextBidder(i)} \times |s|$. If i where to bid $< s, v' >$ such that $v' < AverageCost_{NextBidder(i)} \times |s|$ than he would lose the bid since $\frac{v'}{|s|} < AverageCost_{NextBidder(i)}$ and thus be rated below $NextBidder(i)$ in L. Thus the payment of i is equal to the critical value of i.

□

11.5.4 Examples

1. Let us return to the example we used in 11.5.1, but this time for For G_i with GPS:

Player	$< s_i, v_i >$	$\frac{v_i}{\lvert s_i \rvert}$	t_i
R	$(\{a\}, 10)$	10	9.5
G	$(\{a, b\}, 19)$	9.5	0
B	$(\{b\}, 8)$	8	0

We see the algorithm granted R his bid with a payment of 9.5 which is G's average value, G's bid is denied since some of his goods where allocated to R. B's bid is granted as well with no payment since there is no next player after him in L.

2. Another example of this algorithm at work:

Player	$< s_i, v_i >$	$\frac{v_i}{\lvert s_i \rvert}$	t_i
R	$(\{a\}, 20)$	20	0
G	$(\{b\}, 15)$	15	0
B	$(\{a, b\}, 10)$	10	0

R and G's bids are granted, B's bid is denied. Had R not participated G's bid would still be granted a and B's bid would still be denied, thus his payment is 0. Had G not participated, B's bid would still be denied, thus his payment is 0. In this case the allocation is also the efficient one.

11.6 Single-Unit Combinatorial Auctions

In a *Single-Unit Combinatorial Auction* bidders are interested in buying as many copies of the a single good as offered by the seller. In this case the term *auction* maybe a bit misleading, since the sellers acts more like a shopkeeper that chooses a price tag for the product he is selling without knowing the potential buyers' valuations.

11.6.1 Definitions

1. All buyer valuations of the good are within a given range, thus:

$$\forall i,\ v_i \in [1, w].$$

2. The highest valuation among buyers is denoted by

$$v^* = \max(v_i)$$

11.6.2 Single Copy Auction

In this type of auction only one copy of the good is sold. We construct an algorithm ALG to determine the price tag we will give the product as follows (we are interested, of course, in selling the product for the maximal bidder valuation):

We pick a price tag of 2^i $(0 \leq i \leq \log w)$ with probability of $\frac{1}{\log w}$. We define l such that:

$$2^{l-i} \leq v^* \leq 2^l$$

Effectively we cut the potential tag range into about $\log w$ segments, each segment being twice as wide as the segment preceding it. We randomly choose one of the segments with equal probability and fix the price to be in this segment. OPT, knowing all valuations, will, of course, select a price tag of v^*. Our ALG has $\frac{1}{\log w}$ chance of picking a price tag in the segment containing v^*, a price tag in this segment can be at the worst case equal to $v^*/2$. Thus the expected revenue generated by ALG is bounded from below by $2 \log w$. Thus we get a competitive ratio of:

$$\frac{v^*}{ALG} \leq 2 \log w.$$

11.6.3 Several Copies Auction

Assume several copies of the single product are for sale and they number $\log w$. OPT will always sell all the products for a total revenue of $v^* \log w$ (selling all the products to the buyer with the highest valuation).

Our algorithm, ALG, begins by selling the good with a price of 1 and after every sale we make, we double the price.

We consider the final price tag 2^l, that is the price tag where no willing buyers are left for the product or we run out of stock, and observe two cases:

1. If $2^l \leq v^*$, (actually it is exactly $2^l = v^*$, since this is the only possible way the seller can clear his stock), then the best price we got is no worse than v^*, yielding a competitive ratio of about $\log w$.

2. If $v^* < 2^l$, then exists player j that bought at 2^{l-1}, and so $2^{l-1} \leq v_j \leq v^*$. Thus, the last item sold guarantees the following:

   $$v(ALG) \geq v_j \geq \frac{1}{2} v^*$$

   and since

   $$v(OPT) \leq v^* \log w.$$

   In this case we get a competitive ratio of:

   $$\frac{v(OPT)}{v(ALG)} \leq 2 \log w.$$

11.7 Multi-Unit Combinatorial Auctions

In this part we study multi-unit combinatorial auctions. In a *Multi-Unit Combinatorial Auction* there are n types of goods, for each good i there are k_i copies for sale. We isolate our treatment to auctions where the number of copies of each good are relatively small.

11.7.1 Definitions

- Let U be the set of all possible bundles, thus every member of U is a bundle that may be requested by one of the bidders. Formally:

$$U = \{0, ..., k_1\} \times ... \times \{0, ..., k_n\}$$

- For each bidder j, there exists a valuation function:

$$v_j : U \longrightarrow \Re^+$$

- There exists a lower bound α and and upper bound β. Each bidder desires 0 or at least αk_i and at most βk_i units of good i .

- We simplify the problem by assuming 1 unit exists for each product but players can request fractional amounts of it (bids for each product are in the range $[0, ..1]]$).

- *Demand Oracle* . A demand oracle for valuation v accepts as input a vector of item prices $p_{(1)}, p_{(2)}...p_{(n)}$ and outputs the demand for the items at these prices, i.e. it outputs the vector $\lambda = (\lambda_{(1)}, \lambda_{(2)}, ..., \lambda_{(n)})$ that maximizes the surplus $v(\lambda) - \langle \vec{\lambda}, p \rangle = v(\lambda_{(1)}, \lambda_{(2)}, ..., \lambda_{(n)}) - \sum_i \lambda^{(i)} P^{(i)}$

- *Allocation.* An allocation is a collection of m non-negative vectors $\lambda_1, \lambda_2..., \lambda_m$, where $\lambda_j^{(i)}$ specifies the amount of good i that bidder j has received. An allocation is feasible if for all i, $\sum_j \lambda_j^{(i)} \leq 1$.

- *Value of an allocation.* The value of an allocation A is $V(A) = \sum_j v_j(\lambda_j)$. An allocation is optimal if it achieves the maximal value of any feasible allocation.

- *Direct Relevation mechanism.* A direct relevation mechanism recieves as input a vector of declared valuations $v_1, .., v_m$ and produces as output an allocation $\lambda_1, .., \lambda_m$ and a vector of payments $P_1, .., P_m$, where bidder j recieves λ_j and pays P_j.

- *Incentive Compatibility.* A direct relevation mechanism is inncentive compatible if for every bidder j, every valuation v_j, all declarations of the other bidders v_{-j}, and all possible "false declaratrions" v'_j we have that bidder j's utility with bidding v'_j is no

more than his utility truthfully bidding v_j. I.e. Let λ_j and P_j be the mechanism's output with input (v_j, v_{-j}) and λ'_j and P'_j be the mechanism's output with input (v'_j, v_{-j}) then $v_j(\lambda) - P_j \geq v_j(\lambda') - P'_j$

11.7.2 The Online Algorithm

We present an *Online Algorithm* for the problem of *Multi-Unit Auction with Bounded Demand*. The idea of the algorithm is as follows :
At any point in time good i has a price of $P^{(i)}$. The bidders arrive one after the other, and when bidder j is considered he chooses which bundle he prefers according to the current prices. The prices $P^{(i)}$ are initialized to some parameter P_0 and are increased whenever a quantity of that good is allocated. The increase in price is exponential with a rate r per unit allocation.

Formally, the online Algorithm with Parameters P_0 and r is as follows,

- for each good i, $l_1^{(i)} = 0$
- for each bidder $j = 1$ to m
  - for each good i, $P_j^{(i)} = P_0 r^{l_j^{(i)}}$
  - Query j's demand oracle on the current prices and allocate: $Demand(P_j^{(1)}, ..., P_j^{(n)}) \longrightarrow (x_j^{(1)}, ..., x_j^{(n)})$
  - determine bidder j's payment as: $P_j^{total} = \sum_i x_j^{(i)} P_j^{(i)}$
  - update $l_{j+1}^{(i)} = l_j^{(i)} + x_j^{(i)}$

11.7.3 Analysis of Online Algorithm

The correctness of the algorithm involves three elements: incentive compatibility (as defined in previous lectures), validity and approximation ratio.

Lemma 11.15 *For any outer choice of parameters P_0 and r, the online algorithm is incentive compatible.*

The lemma follows from the theorem below:

Theorem 11.16 *A direct relevation mechanism is incentive compatible if and only if for every bidder j and every vector of bids of the other players v_{-j} it:*

1. *fixes a price $p_j(\lambda)$ for every possible allocation λ to bidder j, and whenever bidder j is allocated λ his payment is $p_j(\lambda)$. (Note that $p_j(\lambda)$ does not depend on v_j.)*

2. *allocates to j, λ that maximizes the value of $v_j(\lambda) - p_j(\lambda)$ over all λ that can be allocated to j (for any choice of v_j).*

Proof:

1. We show the two conditions are sufficient. Fix v_{-j} and v_j. Now consider an alternative "lie" v'_j for bidder j. Let λ and p be the mechanism's output for j with input (v_j, v_{-j}) and λ' and p' be the mechanism's output for for j with input (v'_j, v_{-j}). If $\lambda = \lambda'$ then the first condition ensures that $p = p' = p_j(\lambda)$, and thus both allocation and the payments with declaration v'_j are equivalent to those obtained with a truthful bid. If $\lambda \neq \lambda'$, then $p = p_j(\lambda)$, $p' = p_j(\lambda')$, and the second codition ensures that $v_j(\lambda) - p_j(\lambda) \geq v_j(\lambda') - p_j(\lambda')$, and thus the utility with declaration v'_j is less than that obtained with a truthful bid.

2. We show the two conditions are necessary.

   - Assume to the contrary that the first condition does not hold, i.e. that for some v_{-j}, and the valuations v_j and v'_j, the mechanism yields the same allocation λ to player j, but charges different payments $p > p'$, respectively, from him. Now it is clear that for the case where bidders' valuations are v_{-j} and v_j, for bidder j to declare v'_j instead of v_j will imorove his utility (since the allocation remains the same, while the payment decreases), contrary to the definition of incentive compatibility.
   - Now assume the first condition holds, but assume to the contrary that the second condtion doen't, i.e. that for some v_{-j} and valuation v_j, the mechanism allocates λ to j with the property that $v_j(\lambda) - p_j(\lambda) < v_j(\lambda') - p_j(\lambda')$, for some λ' that can be allocated to j, e.g. if he bids v'_j. But this exactly says that for the case where bidders' valuations are v_{-j} and v_j, the for bidder j to declare v'_j instead of v_j will improve his utility (since he is now allocated λ' and charged $p_j(\lambda')$), contrary to the definition of incentive compatibility.

□

Next we prove the validity of the algorithm. i.e. that it never allocates more than the available quantity of each good. This is true as long as the values of P_0 and r satisfy a certain condition. Let $l_j^{(i)} = \sum_{t=1}^{j-1} x_t^{(i)}$ the total allocation of good i to players in $[1..j-1]$. Let $l_*^{(i)} = l_{m+1}^{(i)}$ the total allocation to all players, $l_*^{(i)} \leq 1$. Let $v_{max} = \max_{j,\lambda} v_j(\lambda)$ be the highest valuation in the auction.

Lemma 11.17 *Let P_0, r be such that the condition $P_0 r^\gamma \geq \frac{v_{max}}{\alpha}$ holds, then $l_*^{(i)} \leq \gamma + \beta$ In particular for $\gamma = 1 - \beta$ the algorithm is valid.*

Proof: Assume to the contrary that $l_{j+1}^{(i)} > \gamma+\beta$, and let j be the first player that caused this to happen for some good i, i.e. $l_{j+1}^{(i)} > \gamma + \beta$ since no player is allocated more than β units of each good, we have that $(l_j^{(i)} > \gamma)$. It follows that $(P_j^{(i)} > P_0 r^{\gamma} \geq \frac{v_{max}}{\beta})$. Since player j is allocated at least α units of good i, his payment is at least $\alpha P_j^{(i)} > v_{max} \geq v_j(x_j)$. Thus player j's payment is more than his valuation for the bundle allocated, in contradiction to the definition of the demand oracle and the possibility of choosing the empty bundle and paying nothing. □

Our final step is to prove a bound on the approximation ratio. For an allocation algorithm A, let V(A) denote the total sum of bidders' valuations for the allocation produced, i.e. $V(A) = \sum_j v_j(x_j)$, where $(x_1, ..., x_m)$ is the allocation produced by A.

We now prove that:

$$V(ALG)(1 + \frac{r^{\beta} - 1}{\beta}) \geq V(OPT) - nP_0.$$

To get this conjecture we prove some additional lemmas.

Lemma 11.18 *For any j and $\overrightarrow{\lambda_j}$, $v_j(\overrightarrow{x_j}) \geq v_j - \langle \lambda_j, \overrightarrow{P_*} \rangle$, where P_* is the vector of the goods' prices at the end of allocation, $P_* = P_*^{(1)} ... P_*^{(n)}$, and where $P_*^{(i)} = P^{(0)} r^{l_*^{(i)}}$*

Proof: When bidder j is allocated than the inequality

$$v_j(\overrightarrow{x_j}) - \langle \overrightarrow{x_j}, \overrightarrow{P_j} \rangle \geq v_j(\overrightarrow{\lambda_j}) - \langle \overrightarrow{\lambda_j}, \overrightarrow{P_j} \rangle$$

takes place. It derives from definition of demand oracle.
Since $\overrightarrow{P_*} \geq \overrightarrow{P_j}$ for any j, then

$$v_j(\overrightarrow{x_j}) - \langle \overrightarrow{x_j}, \overrightarrow{P_j} \rangle \geq v_j(\overrightarrow{\lambda_j}) - \langle \overrightarrow{\lambda_j}, \overrightarrow{P_*} \rangle.$$

The last inequality holds true since $\overrightarrow{P_*} \geq \overrightarrow{P}$. Since $\langle \overrightarrow{x_j}, \overrightarrow{P_j} \rangle \geq 0$ the lemma holds. □

Corollary 11.19

$$V(ALG) \geq V(OPT) - \sum_i P_*^i$$

Since each bidder pays no more than the value of the bundle he gets, the total revenue is a lower bound for the total valuation. When the j is allocated we have

$$v_j(\overrightarrow{x_j})) \geq \langle \overrightarrow{x_j}, \overrightarrow{P_j} \rangle = \sum_i x_j^{(i)} P_{(0)} r^{l_j^{(i)}}$$

Summing for all bidders we have

$$V(ALG) = \sum_j v_j(x_j) \geq \sum_j \sum_i x_j^{(i)} P_0 r^{l_j^{(i)}} = \sum_i \sum_j x_j^{(i)} P_0 r^{l_j^{(i)}}$$

Let $R^{(i)} = \sum_j x_j^{(i)} P_0 r^{l_j^{(i)}}$ be the total revenue obtained for good i. Let $\Delta_j R^{(i)} = x_j^{(i)} P_0 r^{l_j^{(i)}}$, then $R^{(i)} = \sum_j \Delta_j R^{(i)}$. We denote $h = x_j^{(i)}, t = l_j^{(i)}$, so $\Delta_j R^{(i)} = \sum \Delta_j R^{(i)} = h P_0 r^t$.
Let $\Delta\overline{R^{(i)}}$ be the change when the price grows continuously. We compare this value to

$$\frac{\Delta\overline{R^{(i)}}}{P_0} = \int_t^{t+h} r^x dx = \frac{r^x}{\ln r}\Big|_t^{t+h} = \frac{r^{t+h} - r^t}{\ln r} = \frac{r^t}{\ln r}(r^h - 1).$$

Since the demand of any good is bounded by β we can bound the ratio between $\Delta\overline{R^{(i)}}$ and $\Delta R^{(i)}$, (in other words, bounding the ratio between the continuous and discrete evaluations).

$$\max_{h \leq \beta} \frac{\Delta\overline{R^{(i)}}}{\Delta R^{(i)}} = \max_{h \leq \beta} \frac{\frac{r^t}{\ln r}(r^h - 1)}{h r^t} = \max_{h \leq \beta} \frac{r^h - 1}{h \ln r} = \frac{1}{\ln r} \frac{r^\beta - 1}{\beta}$$

And so

$$R^{(i)} \geq \frac{\beta}{r^\beta - 1}(P_*^i - P_0),$$

where $P_* = r^{t+h}$ and $P_0 = r^t$.
Summing this result over all goods, we achieve the following bound:

Lemma 11.20

$$V(ALG) \geq \sum_j R^{(i)} \geq \frac{\beta}{r^\beta - 1}(\sum P_*^{(i)} - nP_0)$$

$$V(ALG) = \frac{r^\beta - 1}{\beta} + nP_0 \geq \sum_i P_*^{(i)}$$

$$V(ALG) \geq V(OPT) - \sum P_*^{(i)}$$

We obtain compatible, valid and approximation algorithm as long as following two conditions on the parameters P_0 and r hold:

1. $nP_0 \leq \frac{V(OPT)}{2}$
2. $r^{1-\beta} \geq \frac{v_{max}}{\alpha P_0}$.

And so

$$V(ALG)\frac{r^{\beta}-1}{\beta} \geq \frac{1}{2}V(OPT).$$

Under these conditions no item is over allocated and the approximation ratio is $C = 2(1 + \frac{r^{\beta}-1}{\beta})$

In order to obtain a complete online algorithm we need to choose parameters to our ALG. In our algorithm we choose them before any players arrive. This is possible, only if there exists an a priory known bounds v_{min} and v_{max} such that:

$$v_{min} \leq \max_j v_j(\beta, ..., \beta) \leq v_{max}$$

We will assume this condition holds.

Using the algorithm with $P_0 = \frac{v_{min}}{2n}$ and $r = \frac{v_{max}}{\alpha P_0}$ we achieve $2\frac{\frac{\rho}{2\alpha n}^{\frac{\beta}{1-\beta}}-1}{\beta}$ - approximation to the optimal allocation, where $\rho = \frac{v_{max}}{v_{min}}$.

11.8 References

1. Mansour Y. "Topics in Computational Game Theory" Course, Notes from lecture 11 presented at 11/6/2004, Tel Aviv University, Israel.

2. Lehmann D, O'Callaghan LI, Shoham Y. Truth Revelation in Approximately Efficient Combinatorial Auctions. Journal of the ACM, October 19, 2002.

3. Bartal Y, Gonen R, Nisan N. Incentive Compatible Multi Unit Combinatorial Auctions. ACM Press, 2003.